AF/ $

IN THE U.S. PATENT AND TRADEMARK OFFICE

In re U.S. Patent Application of:

SERIAL NO.	:	10/589,155
APPLICANT	:	Pedersen et al.
FILING DATE	:	May 4, 2007
ART UNIT	:	2181
EXAMINER	:	Lee, Chun Kuan
DOCKET NO.	:	884A.0148.U1(US)
CUSTOMER NO.	:	29683
TITLE	:	A METHOD FOR CONFIGURING AN ELECTRONIC DEVICE

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S APPEAL BRIEF

Sir:

Commensurate with the Notice of Appeal filed on December 26, 2008, Applicant (hereinafter referred to as "Appellant") hereby submits this Appeal Brief to the Board of Patent Appeals and Interferences (hereinafter, the Board) under 37 C.F.R. §41.31 and §41.37. Please charge Deposit Account No. 50-1924 for the $540 Appeal Brief fee set forth in 37 C.F.R. §41.20(b)(2).

Appellant petitions for a one month extension of time in which to file this Appeal Brief and authorizes the USPTO to charge Deposit Account No. 50-1924 in the amount of $130 for the petition fee. No additional petition or fee is believed to be due. However, should the undersigned attorney be mistaken, please consider this as a petition for any extension of time under 37 C.F.R. §1.136(a) or (b) that may be required to avoid dismissal of this appeal and maintain the pendency of this application, and charge Deposit Account No. 50-1924 for any required fees.

(1) REAL PARTY IN INTEREST

The real party in interest (RPI) is Nokia Corporation of Espoo, Finland, cited in an assignment of the US application recorded on May 4, 2007 at reel 019266, frame 0880.

(2) RELATED APPEALS AND INTERFERENCES

There are no other pending appeals or interferences of which the undersigned attorney and assignee/RPI is aware that will directly affect, be directly affected by or have a bearing on the Board's decision in this appeal.

(3) STATUS OF CLAIMS

Claims 1, 3-10, 12-28, 34-41 and 43-44 stand finally rejected by the Final Office Action dated August 29, 2008 ("the Final Office Action"). These claims are pending in this Appeal, and are reproduced in an Appendix (Section 8) accompanying this Appeal Brief. Claims 2, 11, 29-33 and 42 were previously canceled.

(4) STATUS OF AMENDMENTS

An amendment to the claims was proposed subsequent to the final rejection of the claims in the Final Office Action (see Amendment in Response to Office Action, dated October 24, 2008). In an Advisory Action mailed on November 25, 2008 the Examiner entered the amendments proposed in the afore-referenced Amendment in Response to Office Action. The claims reproduced in the Appendix (Section 8) accompanying this Appeal Brief include the amendments proposed in the afore-referenced Response to Office Action. In particular, "identified" was deleted from lines 4 and 8 of independent claim 1. The word "an" was changed to - -the- - in line 2 of claim 16. The word "a" was changed to - -the- - in line 3 of claim 21. The word "the" was added after "on" in line 3 of claim 34. The spelling of "hierarchical" was corrected in line 3 of claim 43. Lastly, the word "a" was changed to - -the- - in line 9 of claim 44.

(5) SUMMARY OF CLAIMED SUBJECT MATTER

Claims 1, 3-10, 12-28, 34-41, 43 and 44 are pending. Of those claims, claims 1, 17-19, 22, 24, 34-37, 40-41 and 43-44 are independent. Claim 34 includes means plus function elements, as

described in further detail below.

More particularly, independent claim 1 recites a method comprising: receiving at an electronic device a command specifying execution of an unidentified executable on first data; automatically determining, from metadata of the first data, a content type of the first data; automatically identifying an executable using the content type determined from the metadata; and operating on the first data using the identified executable.

Independent claim 17 recites a method, comprising: transferring code comprising a command to an electronic device, wherein the command specifies execution of an unidentified executable on first data stored at a first leaf node of a hierarchical nodular data structure; determining, from metadata of the first leaf node, a content type of the first data; identifying an executable using the content type determined from the metadata of the identified first leaf node; and operating on the first data, stored at the identified first leaf node, using the identified executable.

Independent claim 18 recites a method, comprising: receiving re-usable code at an electronic device wherein the code comprises: commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and a further command specifying execution of an unidentified executable on the first data stored at the first leaf node; determining, from metadata stored at the first leaf node, a content type of the first data stored at the first leaf node; identifying an executable using the content type determined from the metadata stored at the first leaf node; and operating on the first data stored at the first leaf node using the identified executable.

Independent claim 19 recites an electronic device, comprising: a memory configured to store first data and metadata of the first data; a receiver configured to receive a command specifying execution of an unidentified executable on the first data; and a processor configured to determine from the metadata of the first data, a content type of the first data, to identify an executable using the content type determined from the metadata, and to operate on the first data using the identified executable.

Independent claim 22 recites a data structure embodied on a computer-readable medium, comprising: code identifying first data and specifying execution of an unidentified executable on the first data.

Independent claim 24 recites a data structure embodied on a computer-readable medium, comprising: commands, execution of which create at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and a further command identifying the first leaf node and specifying execution of an unidentified executable on the first data stored at the first leaf node.

Independent claim 34 recites an electronic device, comprising: means for storing first data; means for receiving a command specifying execution of an unidentified executable on the first data; means for determining, from metadata, a content type of the identified first data; means for identifying an executable using the content type determined from the metadata; and means for operating on the identified data using the identified executable.

Independent claim 35 recites a method, comprising: providing code identifying first data and specifying execution of an unidentified executable on the first data and transmitting the code.

Independent claim 36 recites a method, comprising: transmitting commands for creating a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and transmitting a further command specifying execution of an unidentified executable on the first data stored at the first leaf.

Independent claim 37 recites a server, comprising: a memory configured to store code identifying first data and specifying execution of an unidentified executable on the first data; and an interface configured to transmit the code.

Independent claim 40 recites a server, comprising: a memory configured to store commands, execution of which resulting in creation at an electronic device, of a hierarchical nodular data

structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node, and configured to store a further command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first leaf node; and a transmitter configured to transmit the stored instructions.

Independent claim 41 recites a computer program product comprising program instructions embodied on a tangible computer-readable medium, execution of the program instructions resulting in operations comprising: automatically determining, from metadata of first data, a content type of first data; automatically identifying an executable using the content type determined from the metadata; and enabling the first data to be operated on using the identified executable.

Independent claim 43 recites a method, comprising: receiving a first command at an electronic device, the first command specifying creation of a leaf node in a hierarchical data structure, and identifying first data to be stored at the leaf node and metadata indicating a content type of the first data; creating the leaf node at the electronic device; receiving a second command, at the electronic device, that specifies execution of an unidentified executable on the first data stored at the created leaf node; determining, from the metadata, the ~~a~~ content type of the first data; identifying an executable using the content type determined from the metadata; and operating on the first data using the identified executable.

Lastly, independent claim 44 recites an electronic device, comprising: a receiver configured to receive a first command at an electronic device, the first command specifying creation of a leaf node in a hierarchical data structure, and identifying first data to be stored at the leaf node and metadata indicating a content type of the first data; and a processor configured to create the leaf node at the electronic device, wherein the receiver is further configured to receive a second command that specifies execution of an unidentified executable on the first data stored at the created leaf node, and the processor is further configured to determine, from the metadata, the content type of the first data, to identify an executable using the content type determined from the

metadata, and to operate on the first data using the identified executable.

It is respectfully noted that in making reference herein to various portions of the specification and drawings (pursuant to 37 CFR §41.37(c)(1)(v)), Appellant does not intend to limit the claims in any way. All references to the page and line numbers of the specification, as well as reference to the drawings, are for illustrative purposes and provide non-limiting examples and embodiments, which should not be understood to limit the scope of the claims in any way.

For example, embodiments of the claimed subject matter are directed to configuring electronic devices, in particular, mobile cellular telephones (page 1, lines 1 to 9). The claimed methods, electronic devices, data structures, servers and computer program may be advantageous, for example, for a set-up process used for more than one electronic device without specific adaptation for each device (page 1, line 33 – page 2, line 1). Some embodiments relate to configuring the electronic devices using a hierarchical nodular data structure, for example, in SyncML (page 2, lines 5 to 7; page 6, lines 17 to 30).

Previously, a user of a mobile phone would have spent a considerable amount of time and effort configuring a mobile phone's settings so that it works correctly and/or as they would wish and therefore a new mobile phone may appear less attractive to a user due to the large amount of time and effort required (page 1, lines 16 to 19). By providing a configuration process that does not require specific adaptation for each device, the time and effort required to configure a mobile phone may be reduced (page 3, lines 9 to 11).

Independent claim 1 is directed to a method whereby an electronic device receives a command (page 3, lines 19 to 20). The command may specify execution of an unidentified executable on first data (page 4, lines 20 to 21; page 10, lines 11 to 13), i.e. commands "do something on that" – it specifies 'that' but not 'something'. The command may be sent from, for example, a mobile phone service provider or server (i.e. 'sending client'), and received by, for example, a mobile phone (i.e. receiving client) (page 1, lines 25 to 26; page 7, lines 15 to 18). A Device Manager (DM) client within, for example, a mobile phone, may access

identified first data and read the properties of metadata to determine the content type of the first data (page 10, line 29 to page 11, line 1). The metadata may include, for example, the content of the format element and/or the content of the type element of the first data. The executable is identified using the content type of the first data and this executable is used to operate on the first data (page 11, lines 3 to 13). For example, the content type may be a sound file, video file, picture file, Java Midlet, contact details, or a bookmark, and the executable may be an audio player, video player, picture viewer, Java Virtual Machine, an executable that adds the contact details to a telephone's contact list, or an executable that adds the bookmark to the phone's bookmark list, respectively (page 11, lines 3 to 13).

Independent claim 17 is directed to a method whereby code comprising a command is transferred to an electronic device (page 2, lines 5 to 7). The command specifies execution of an unidentified executable on first data stored at a first leaf node of a hierarchical nodular data structure (page 3, lines 26 to 28; page 10, lines 11 to 15), i.e. commands "do something on that" – it specifies 'that' but not 'something'. A Device Manager (DM) client within, for example, a mobile phone, may access identified first data and read the properties of metadata to determine the content type of the first data stored at the first leaf node (page 10, line 29 to page 11, line 1). The metadata may include, for example, the content of the format element and/or the content of the type element of the first data. The executable is identified using the content type of the first data and this executable is used to operate on the first data (page 11, lines 3 to 13). For example, the content type may be a sound file, video file, picture file, Java Midlet, contact details, or a bookmark, and the executable may be an audio player, video player, picture viewer, Java Virtual Machine, an executable that adds the contact details to a telephone's contact list, or an executable that adds the bookmark to the phone's bookmark list, respectively (page 11, lines 3 to 13). For example, the code may be set-up code comprising a portion for carrying out executables (page 7, line 33 – page 8, line 3).

Independent claim 18 is directed to a method whereby re-usable code comprising commands is received by an electronic device (page 3, lines 13 to 14). For example, the code may be comprised in two portions: a first portion for creating a management tree; and a second

portion for carrying out executables (page 7, line 33 to page 8, line 3). Thus, commands may be used to create a hierarchical nodular data structure at the electronic device, and further commands may be used to specify execution of an unidentified executable on first data stored at a leaf node of the created hierarchical nodular data structure (page 3, lines 26 to 28; page 6, lines 23 to 30; page 8, lines 1 to 2; page 10, lines 11 to 15). A Device Manager (DM) client within, for example, a mobile phone, may access identified first data and read the properties of metadata to determine the content type of the first data stored at the first leaf node (page 10, line 29 to page 11, line 1). The metadata may include, for example, the content of the format element and/or the content of the type element of the first data. The executable is identified using the content type of the first data and this executable is used to operate on the first data (page 11, lines 3 to 13). For example, the content type may be a sound file, video file, picture file, Java Midlet, contact details, or a bookmark, and the executable may be an audio player, video player, picture viewer, Java Virtual Machine, an executable that adds the contact details to a telephone's contact list, or an executable that adds the bookmark to the phone's bookmark list, respectively (page 11, lines 3 to 13). For example, the code may be set-up code comprising a portion for carrying out executables (page 7, line 33 – page 8, line 3). As the executable specified by the code is "unidentified", the code can be used with multiple different devices without adaptation for each device, and is therefore "re-usable" (page 7, lines 15 to 16).

Independent claim 19 is directed to an electronic device comprising a memory, receiver and a processor (page 5, lines 26 to 27). The memory may store first data and metadata of the first data (page 6, lines 17 to 18). The receiver may receive a command specifying execution of an unidentified executable on first data (page 4, lines 20 to 21; page 10, lines 11 to 15). The processor may determine the content type of the first data from metadata (page 10, line 29 – page 11, line 1). The processor may access identified first data and read the properties of metadata to determine the content type of the first data stored at the first leaf node (page 10, line 29 to page 11, line 1). The metadata may include, for example, the content of the format element and/or the content of the type element of the first data. The executable is identified using the content type of the first data and this executable is used to operate on the first data

(page 11, lines 3 to 13). For example, the content type may be a sound file, video file, picture file, Java Midlet, contact details, or a bookmark, and the executable may be an audio player, video player, picture viewer, Java Virtual Machine, an executable that adds the contact details to a telephone's contact list, or an executable that adds the bookmark to the phone's bookmark list, respectively (page 11, lines 3 to 13). For example, the code may be set-up code comprising a portion for carrying out executables (page 7, line 33 – page 8, line 3).

Independent claim 22 is directed to a data structure embodied on a computer-readable medium (page 7, lines 30 to 31). The data structure may comprise code that identifies first data and specifies execution of an unidentified executable on the first data (page 4, lines 20 to 21; page 10, lines 11 to 15). For example, the code may be set-up code as illustrated in Figure 2 (page 7, lines 30 to 31). The set-up code may be contained within a smart card (page 7, lines 9 to 11).

Independent claim 24 is directed to a data structure embodied on a computer-readable medium (page 7, lines 30 to 31). For example, the data structure may be comprised in two portions: a first portion with commands for creating a management tree; and a second portion with commands for carrying out executables (page 7, line 33 to page 8, line 3). Thus, commands may be used to create a hierarchical nodular data structure at the electronic device, and further commands may be used to specify execution of an unidentified executable on first data stored at a leaf node of the created hierarchical nodular data structure (page 3, lines 26 to 28; page 6, lines 23 to 30; page 8, lines 1 to 2; page 10, lines 11 to 15).

Independent claim 34 is directed to an electronic device comprising a means for storing, means for receiving, means for determining, means for identifying, and means for operating. The means for storing may be a memory (page 5, lines 26 to 27; figure 1, reference numeral 13). The means for receiving may be a cellular radio transceiver (page 5, lines 26 to 27; figure 1, reference numeral 12). The means for storing, determining, identifying and operating may be a processor (page 5, lines 26 to 27; figure 1, reference numeral 11). The electronic device may store first data and receive a command specifying execution of an

unidentified executable on the first data (page 6; lines 17 to 18; page 10, lines 11 to 13; page 4, lines 20 to 21; page 10, lines 11 to 13), i.e. commands "do something on that" – the command specifies 'that' but not 'something'. The command may be sent from, for example, a mobile phone service provider or server (i.e. 'sending client'), and received by, for example, a mobile phone (i.e. receiving client) (page 1, lines 25 to 26; page 7, lines 15 to 18). A Device Manager (DM) client within, for example, a mobile phone, may access identified first data and read the properties of metadata to determine the content type of the first data (page 10, line 29 to page 11, line 1). The metadata may include, for example, the content of the format element and/or the content of the type element of the first data. The executable is identified using the content type of the first data and this executable is used to operate on the first data (page 11, lines 3 to 13). For example, the content type may be a sound file, video file, picture file, Java Midlet, contact details, or a bookmark, and the executable may be an audio player, video player, picture viewer, Java Virtual Machine, an executable that adds the contact details to a telephone's contact list, or an executable that adds the bookmark to the phone's bookmark list, respectively (page 11, lines 3 to 13).

Independent claim 35 is directed to a method of providing and transmitting code identifying first data (page 3, lines 19 to 20). The code also specifies execution of an unidentified executable on the first data (page 4, lines 20 to 21; page 10, lines 11 to 15). For example, the code may be set-up code as illustrated in Figure 2 (page 7, lines 30 to 31). The set-up code may be contained within a smart card or may be stored at a server for transfer to an electronic device (page 7, lines 9 to 11; page 7, lines 15 to 18).

Independent claim 36 is directed to a method for transmitting commands for configuring an electronic device. The method comprises transmitting commands for creating a hierarchical nodular data structure (page 6, lines 23 to 30; page 8, lines 1 to 2) and commands for specifying execution of an unidentified executable on first data stored at a first leaf node of the hierarchical nodular data structure (page 3, lines 26 to 28; page 10, lines 11 to 15). Thus, commands may be used to create a hierarchical nodular data structure at the electronic device, and further commands may be used to specify execution of an unidentified executable on first

data stored at a leaf node of the created hierarchical nodular data structure (page 3, lines 26 to 28; page 6, lines 23 to 30; page 8, lines 1 to 2; page 10, lines 11 to 15).

Independent claim 37 is directed to a server for storing and transmitting code (page 6, lines 12 to 15; page 7, line 15 – page 8, line 3). The code identifies first data and specifies execution of an unidentified executable on the first data (page 4, lines 20 to 21; page 10, lines 11 to 15). For example, the code may be set-up code as illustrated in Figure 2 (page 7, lines 30 to 31).

Independent claim 40 is directed to a server for storing and transmitting commands (page 6, lines 12 to 15; page 7, line 15 – page 8, line 3). The commands may be used to create a hierarchical nodular data structure at an electronic device (page 6, lines 23 to 30; page 8, lines 1 to 2) and further commands may be used to specify execution of an unidentified executable on first data stored at a first leaf node of the hierarchical nodular data structure (page 3, lines 26 to 28; page 10, lines 11 to 15).

Independent claim 41 is directed to a computer program product comprising computer program instructions embodied on a tangible computer-readable medium (page 4, lines 3 to 6). For example, a mobile phone using the computer program, may execute computer program instructions to access identified first data and read the properties of metadata and determine the content type of the first data (page 10, line 29 to page 11, line 1). The metadata may include, for example, the content of the format element and/or the content of the type element of the first data. The computer program may then identify an executable using the content type of the first data and this executable may then be used to operate on the first data (page 11, lines 3 to 13). For example, the content type may be a sound file, video file, picture file, Java Midlet, contact details, or a bookmark, and the executable may be an audio player, video player, picture viewer, Java Virtual Machine, an executable that adds the contact details to a telephone's contact list, or an executable that adds the bookmark to the phone's bookmark list, respectively (page 11, lines 3 to 13).

Independent claim 43 is directed to a method whereby an electronic device receives

commands from, for example, a server (page 3, lines 19 to 20; page 7, lines 1 to 2; page 7, lines 16 to 18). For example, the commands may be comprised in two portions: a first portion for updating a management tree; and a second portion for carrying out executables (page 7, line 33 to page 8, line 3). Thus, commands may be used to update a hierarchical nodular data structure at the electronic device by specifying creation of a leaf node and identifying first data to be stored at the first leaf node, and further commands may be used to specify execution of an unidentified executable on first data stored at the newly created leaf node (page 3, lines 26 to 28; page 6, lines 23 to 30; page 8, lines 1 to 2; page 10, lines 11 to 15). A Device Manager (DM) client within, for example, a mobile phone, may access identified first data and read the properties of metadata to determine the content type of the first data stored at the first leaf node (page 10, line 29 to page 11, line 1). The metadata may include, for example, the content of the format element and/or the content of the type element of the first data. The executable is identified using the content type of the first data and this executable is used to operate on the first data (page 11, lines 3 to 13). For example, the content type may be a sound file, video file, picture file, Java Midlet, contact details, or a bookmark, and the executable may be an audio player, video player, picture viewer, Java Virtual Machine, an executable that adds the contact details to a telephone's contact list, or an executable that adds the bookmark to the phone's bookmark list, respectively (page 11, lines 3 to 13).

Independent claim 44 is directed to an electronic device. The electronic device receives a first command specifying creation of a leaf node in a hierarchical nodular data structure in the electronic device (page 6, lines 32 to 33; page 8, lines 5 to 7). The electronic device also receives a second command specifying execution of an unidentified executable on first data stored at the created leaf node (page 10, lines 11 to 15). The electronic device may determine the content type of the first data from metadata and identify an appropriate executable (page 10, line 29 – page 11, line 1; page 11, lines 3 to 13). The first data stored at the created leaf node is operated on using the identified executable. For example, the content type may be a sound file, video file, picture file, Java Midlet, contact details, or a bookmark, and the executable may be an audio player, video player, picture viewer, Java Virtual Machine, an executable that adds the contact details to a telephone's contact list, or an executable that

adds the bookmark to the phone's bookmark list, respectively (page 11, lines 3 to 13).

Embodiments of the invention may therefore enable, for example, an executable resident on a target device (receiving client) to be executed as a consequence of a command from a transmitter device (sending client) without a need for the transmitter device to specify the identity of that executable (unidentified at the sending client). As the identity of a resident executable may vary in an unknown way from target device to target device, this allows in some embodiments, SyncML code to be re-used to command performance of a certain common process on a plurality of different target devices (page 4, lines 23 to 28). This may be desirable, for example, for a service provider to initially configure all or some mobile phones (page 1, line 33 to page 2, line 3).

The following table is presented to provide further non-limiting detail and support.

Table

Independent claim 1	**(Page; Line)** – application as filed
A method comprising:	Title (3; 18) (3; 25) (3; 33) Independent claim 1 Independent claim 17 Independent claim 18 Independent claim 30 Figures 1 to 4
receiving at an electronic device a command specifying execution of an unidentified executable on first data;	(3; 19 – 20; independent claim 1) "...receiving at an electronic device a command..." (4; 13; independent claim 19) "...means for receiving a command..." (4; 20 – 21; independent claim 22) "...specifying execution of an unidentified executable on the first data..." (4; 27 – 28; independent claim 24) "...first command ...that specifies execution of an unidentified executable on the first data" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable"

	(10; 11 – 13) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used"
automatically determining, from metadata of the first data, a content type of the first data;	(8; 15 – 17) "<Meta>… <Format>… <Type>" (10; 29 – 11; 1) "The DM client…reads the … properties contained within *meta.* The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data…" Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12
automatically identifying an executable using the content type determined from the metadata; and	(3; 22 – 23; independent claim 1) "automatically identifying an executable from the determined property" (3; 29; independent claim 17) "identifying an executable from the determined property" (4; 7; independent claim 18) "identifying an executable from the determined property" (4; 14 – 15; independent claim 19) "means for identifying an executable from the determined property" (10; 12 – 15) "It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (11; 3 – 13) "The DM client associates possible *Formats* and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player…"
operating on the first data using the identified executable.	(3; 22 – 23; independent claim 1) "operating on the identified first data using the identified executable" (3; 30 – 31; independent claim 17) "operating on data stored at the identified first leaf node using the identified executable" (4; 8 – 9; independent claim 18) "operating on the first data stored at the first leaf node using the identified executable" (4; 15 – 16; independent claim 19) "operating on the identified first data using the identified executable"

Independent claim 17

Independent claim 17	**(Page; Line)** – application as filed
A method, comprising:	Title (3; 18) (3; 25) (3; 33) Independent claim 1 Independent claim 17 Independent claim 18 Independent claim 30 Figures 1 to 4
transferring code comprising a command to an electronic device, wherein the command specifies execution of an unidentified executable on first data stored at a first leaf node of a hierarchical nodular data structure;	(2; 5 – 7) "...using a SyncML message to transfer data to a target device and perform executables on the transferred data at the target device..." (3; 19 – 20; independent claim 1) "...a command ..." (3; 26 – 28; independent claim 17) "...transferring code comprising a command to a mobile cellular telephone, wherein the command identifies a first leaf node of a hierarchical nodular data structure..." (4; 2; independent claim 18) "...transferring re-usable code to a mobile cellular telephone..." (4; 20 – 21; independent claim 22) "...specifying execution of an unidentified executable on the first data..." (4; 25 – 28; independent claim 24) "...a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node..." (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable..." (5; 3 – 4) "Embodiments of the invention may enable an executable resident on a target device to be used on specified data..." (6; 23 – 30) "A management tree is a hierarchical nodular data structure by which the management client interacts with the mobile telephone 10. The MC may store or retrieve values from the tree and manipulate the properties of the tree. The management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child

	(dependent) nodes but can store a value..." (7; 17 – 18) "The DM session 62 is used to transfer the stored set-up code 50 to the mobile device 10" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" Dependent claim 4 Dependent claim 9 Dependent claim 10 Dependent claim 11 Dependent claim 16
determining, from metadata of the first leaf node, a content type of the first data;	(8; 15 – 17) "<Meta>... <Format>... <Type>" (10; 29 – 11; 1) "The DM client...reads the ... properties contained within *meta.* The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data..." Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12
identifying an executable using the content type determined from the metadata of the identified first leaf node; and	(3; 22 – 23; independent claim 1) "automatically identifying an executable from the determined property" (3; 29; independent claim 17) "identifying an executable from the determined property" (4; 7; independent claim 18) "identifying an executable from the determined property" (4; 14 – 15; independent claim 19) "means for identifying an executable from the determined property" (10; 12 – 15) "It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (11; 3 – 13) "The DM client associates possible *Formats* and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player..."
operating on the first data, stored at the identified first leaf node, using the	(3; 22 – 23; independent claim 1) "operating on the identified first data using the identified executable" (3; 30 – 31; independent claim 17) "operating on data stored

identified executable.	at the identified first leaf node using the identified executable" (4; 8 – 9; independent claim 18) "operating on the first data stored at the first leaf node using the identified executable" (4; 15 – 16; independent claim 19) "operating on the identified first data using the identified executable"

Independent claim 18

Independent claim 18	**(Page; Line)** – application as filed
A method, comprising:	Title (3; 18) (3; 25) (3; 33) Independent claim 1 Independent claim 17 Independent claim 18 Independent claim 30 Figures 1 to 4
receiving re-usable code at an electronic device wherein the code comprises:	(3; 13 – 14) "...using SyncML code that can be re-used for other devices..." (4; 2; independent claim 18) "...transferring re-usable code to a mobile cellular telephone..." (4; 18 – 20; independent claim 22) "...a data structure for re-use in setting-up different mobile cellular telephones, comprising... code" (4; 23 – 24; independent claims 24 and 26) "...a data structure for re-use in setting-up different electronic device..." (4; 32 – 33) "...this allows SyncML code to be re-used..." (5; 7) "...a data structure for re-use in setting-up different electronic devices..." (7; 30 – 31) "The set-up code 50 is a data structure for re-use in setting-up different mobile devices..."
commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and	(4; 3 – 5; independent claim 18) "commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (4; 25 – 27; independent claim 24) "commands for creating at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node"

	(6; 23 – 30) "A management tree is a hierarchical nodular data structure by which the management client interacts with the mobile telephone 10. The MC may store or retrieve values from the tree and manipulate the properties of the tree. The management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child (dependent) nodes but can store a value…" (8; 1 – 2) "There is a first portion 52 for creating a management tree or updating an existing management tree" (8; 5 – 7) "The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree" Dependent claim 16 Figure 4
a further command specifying execution of an unidentified executable on the first data stored at the first leaf node;	(3; 19 – 20; independent claim 1) "a command …" (3; 27 – 28) "wherein the command identifies a first leaf node of a hierarchical nodular data structure" (4; 20 – 21; independent claim 22) "specifying execution of an unidentified executable on the first data" (4; 27 – 28; independent claim 24) "a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies"
determining, from metadata stored at the first leaf node, a content type of the first data stored at the first leaf node;	(8; 15 – 17) "<Meta>… <Format>… <Type>" (10; 29 – 11; 1) "The DM client…reads the … properties contained within *meta.* The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data…" Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12

identifying an executable using the content type determined from the metadata stored at the first leaf node; and	(3; 22 – 23; independent claim 1) "automatically identifying an executable from the determined property" (3; 29; independent claim 17) "identifying an executable from the determined property" (4; 7; independent claim 18) "identifying an executable from the determined property" (4; 14 – 15; independent claim 19) "means for identifying an executable from the determined property" (10; 12 – 15) "It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (11; 3 – 13) "The DM client associates possible *Formats* and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player..."
operating on the first data stored at the first leaf node using the identified executable.	(3; 22 – 23; independent claim 1) "operating on the identified first data using the identified executable" (3; 30 – 31; independent claim 17) "operating on data stored at the identified first leaf node using the identified executable" (4; 8 – 9; independent claim 18) "operating on the first data stored at the first leaf node using the identified executable" (4; 15 – 16; independent claim 19) "operating on the identified first data using the identified executable"

Independent claim 19

Independent claim 19	**(Page; Line)** – application as filed
An electronic device, comprising:	Title (1; 7) "...an electronic device..." (1; 30 to 2; 1) "It is therefore desirable to provide some form of automatic set-up process for electronic devices and, in particular, mobile cellular telephones..." (3; 18 – 19) "...an electronic device..."
a memory configured to store first data and metadata of the first data;	(5; 26 – 27) "mobile cellular telephone, comprises... a memory 13" (6; 17 – 18) "In the mobile telephone 10, processor 11 operates

	as a management client (MC) and can maintain a management tree data structure 100 in the memory 13" Figure 1 Figure 2
a receiver configured to receive a command specifying execution of an unidentified executable on the first data; and	(3; 19 – 20; independent claim 1) "...receiving at an electronic device a command..." (4; 13; independent claim 19) "...means for receiving a command..." (4; 20 – 21; independent claim 22) "...specifying execution of an unidentified executable on the first data..." (4; 27 – 28; independent claim 24) "...first command ...that specifies execution of an unidentified executable on the first data" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (5; 3 – 4) "Embodiments of the invention may enable an executable resident on a target device to be used on specified data" (5; 26 – 27) "mobile cellular telephone, comprises... a cellular radio transceiver 12" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies"
a processor configured to determine from the metadata of the first data, a content type of the first data, to identify an executable using the content type determined from the metadata, and to operate on the first data using the identified executable	(5; 26 – 27) "mobile cellular telephone, comprises: a processor 11" (6; 17 – 21) "In the mobile telephone 10, processor 11 operates as a management client (MC) and can maintain a management tree data structure 100 in the memory 13. The MC correctly interprets SyncML DM commands received from the server, executes appropriate actions in the mobile telephone 10 and sends back relevant responses to the issuing management server via the transceiver 12" (10; 17 – 18) "The DM client of the mobile cellular telephone processes the received set-up code 50" (8; 15 – 17) "<Meta>... <Format>... <Type>" (10; 29 – 11; 1) "The DM client...reads the ... properties contained within *meta.* The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data..." Dependent claim 2

	Dependent claim 10 Dependent claim 11 Dependent claim 12 (3; 22 – 23; independent claim 1) "automatically identifying an executable from the determined property" (3; 29; independent claim 17) "identifying an executable from the determined property" (4; 7; independent claim 18) "identifying an executable from the determined property" (4; 14 – 15; independent claim 19) "means for identifying an executable from the determined property" (10; 12 – 15) "It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (11; 3 – 13) "The DM client associates possible *Formats* and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player..." (3; 22 – 23; independent claim 1) "operating on the identified first data using the identified executable" (3; 30 – 31; independent claim 17) "operating on data stored at the identified first leaf node using the identified executable" (4; 8 – 9; independent claim 18) "operating on the first data stored at the first leaf node using the identified executable" (4; 15 – 16; independent claim 19) "operating on the identified first data using the identified executable"

Independent claim 22

Independent claim 22	**(Page; Line)** – application as filed
A data structure embodied on a computer-readable medium, comprising:	(3; 28) "hierarchical nodular data structure" (4; 3 – 4) "hierarchical nodular data structure" (4; 18 – 21; independent claim 22) "...a data structure..." (4; 23 – 24; independent claim 24) "...a data structure..." (5; 6 – 7; independent claim 26) "...a data structure..."

	(6; 18) "management tree data structure 100" (6; 23) "hierarchical nodular data structure" (7; 9 – 11) "The automatic set-up process may, for example, be initiated by inserting a smart card 16 into the cellular mobile telephone 10. The smart card 16 contains the necessary information to bootstrap the automatic set-up process" (7; 30 – 31) "The set-up code 50 is a data structure for re-use in setting-up different mobile devices" Figure 2
code identifying first data and specifying execution of an unidentified executable on the first data.	(3; 19 – 20; independent claim 1) "...receiving at an electronic device a command..." (4; 13; independent claim 19) "...means for receiving a command..." (4; 20 – 21; independent claim 22) "code identifying first data and specifying execution of an unidentified executable on the first data" (4; 27 – 28; independent claim 24) "...first command ...that specifies execution of an unidentified executable on the first data" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (5; 3 – 4) "Embodiments of the invention may enable an executable resident on a target device to be used on specified data" (7; 33 to 8; 3) "The set-up code 50, in this example, comprises two portions... There is a second portion 54 for carrying out executables." (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (10; 23 – 27) "The portion 54 for carrying out executables is processed as follows. The code is parsed to identify the first sub-portion. The first sub-portion is parsed to identify the URI specified by the element *LocURI* contained within the element *source.* The element LocURI identifies a leaf node within the newly created sub-tree of the management tree" Dependent claim 21

Independent claim 24

Independent claim 24	**(Page; Line)** – application as filed
A data structure embodied on a computer-readable medium, comprising:	(3; 28) "hierarchical nodular data structure" (4; 3 – 4) "hierarchical nodular data structure" (4; 18 – 21; independent claim 22) "...a data structure..." (4; 23 – 24; independent claim 24) "...a data structure..." (5; 6 – 7; independent claim 26) "...a data structure..." (6; 18) "management tree data structure 100" (6; 23) "hierarchical nodular data structure" (7; 9 – 11) "The automatic set-up process may, for example, be initiated by inserting a smart card 16 into the cellular mobile telephone 10. The smart card 16 contains the necessary information to bootstrap the automatic set-up process" (7; 30 – 31) "The set-up code 50 is a data structure for re-use in setting-up different mobile devices" Figure 2
Commands, execution of which create at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and	(4; 3 – 5; independent claim 18) "commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (4; 25 – 27; independent claim 24) "commands for creating at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (6; 23 – 30) "A management tree is a hierarchical nodular data structure by which the management client interacts with the mobile telephone 10. The MC may store or retrieve values from the tree and manipulate the properties of the tree. The management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child (dependent) nodes but can store a value..." (8; 1 – 2) "There is a first portion 52 for creating a management tree or updating an existing management tree" (8; 5 – 7) "The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree" Dependent claim 16

	Figure 4
a further command identifying the first leaf node and specifying execution of an unidentified executable on the first data stored at the first leaf node.	(3; 19 – 20; independent claim 1) "a command ..." (3; 27 – 28) "wherein the command identifies a first leaf node of a hierarchical nodular data structure" (4; 20 – 21; independent claim 22) "specifying execution of an unidentified executable on the first data" (4; 27 – 28; independent claim 24) "a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies"

Independent claim 34

Independent claim 34	**(Page; Line)** – application as filed
An electronic device, comprising:	Title (1; 7) "Embodiments of the present invention relate to configuring an electronic device" (1; 30 to 2; 1) "It is therefore desirable to provide some form of automatic set-up process for electronic devices and, in particular, mobile cellular telephones. It would be desirable for this set-up process to be such that it can be used with more than one electronic device without specific adaptation for that device" (3; 18 – 19) "According to one embodiment of the invention there is provided a method for automatically configuring an electronic device comprising"
means for storing first data;	(5; 26 – 27) "mobile cellular telephone, comprises... a memory 13" (6; 17 – 18) "In the mobile telephone 10, processor 11 operates as a management client (MC) and can maintain a management tree data structure 100 in the memory 13" Figure 1 Figure 2

means for receiving a command specifying execution of an unidentified executable on the first data;	(5; 26 – 27) "mobile cellular telephone, comprises... cellular radio transceiver 12..." (3; 19 – 20; independent claim 1) "...receiving at an electronic device a command..." (4; 13; independent claim 19) "...means for receiving a command..." (4; 20 – 21; independent claim 22) "...specifying execution of an unidentified executable on the first data..." (4; 27 – 28; independent claim 24) "...first command ...that specifies execution of an unidentified executable on the first data" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (10; 11 – 13) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used"
means for determining, from metadata, a content type of the identified first data;	(5; 26 – 27) "mobile cellular telephone, comprises... processor 11..." (8; 15 – 17) "<Meta>... <Format>... <Type>" (10; 29 – 11; 1) "The DM client...reads the ... properties contained within *meta*. The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data..." Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12
means for identifying an executable using the content type determined from the metadata; and	(5; 26 – 27) "mobile cellular telephone, comprises... processor 11..." (3; 22 – 23; independent claim 1) "automatically identifying an executable from the determined property" (3; 29; independent claim 17) "identifying an executable from the determined property" (4; 7; independent claim 18) "identifying an executable from the determined property" (4; 14 – 15; independent claim 19) "means for identifying an executable from the determined property" (10; 12 – 15) "It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (11; 3 – 13) "The DM client associates possible *Formats*

	and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player..."
means for operating on the identified data using the identified executable.	(5; 26 – 27) "mobile cellular telephone, comprises... processor 11..." (3; 22 – 23; independent claim 1) "operating on the identified first data using the identified executable" (3; 30 – 31; independent claim 17) "operating on data stored at the identified first leaf node using the identified executable" (4; 8 – 9; independent claim 18) "operating on the first data stored at the first leaf node using the identified executable" (4; 15 – 16; independent claim 19) "operating on the identified first data using the identified executable"

Independent claim 35

Independent claim 35	**(Page; Line)** – application as filed
A method, comprising:	Title (3; 18) (3; 25) (3; 33) Independent claim 1 Independent claim 17 Independent claim 18 Independent claim 30 Figures 1 to 4
providing code identifying first data and specifying execution of an unidentified executable on the first data; and	(3; 19 – 20; independent claim 1) "...a command identifying first data..." (4; 13; independent claim 19) "...command identifying the first data..." (4; 20 – 21; independent claim 22) "...specifying execution of an unidentified executable on the first data..." (4; 27 – 28; independent claim 24) "...a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node..." (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed

	without specifying the identity of that executable..." (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies..."
transmitting the code.	(6; 5 – 6) "In operation, the processor receives and transmits data via the transceiver 12 and writes and reads data from the memory 13" Dependent claim 25

Independent claim 36

Independent claim 36	**(Page; Line)** – application as filed
A method, comprising:	Title (3; 18) (3; 25) (3; 33) Independent claim 1 Independent claim 17 Independent claim 18 Independent claim 30 Figures 1 to 4
transmitting commands for creating a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and	(4; 3 – 5; independent claim 18) "commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (4; 25 – 27; independent claim 24) "commands for creating at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (6; 23 – 30) "A management tree is a hierarchical nodular data structure by which the management client interacts with the mobile telephone 10. The MC may store or retrieve values from the tree and manipulate the properties of the tree. The management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child (dependent) nodes but can store a value..."

	(8; 1 – 2) "There is a first portion 52 for creating a management tree or updating an existing management tree" (8; 5 – 7) "The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree" Dependent claim 16 Figure 4
transmitting a further command specifying execution of an unidentified executable on the first data stored at the first leaf node.	(3; 19 – 20; independent claim 1) "a command ..." (3; 27 – 28) "wherein the command identifies a first leaf node of a hierarchical nodular data structure" (4; 20 – 21; independent claim 22) "specifying execution of an unidentified executable on the first data" (4; 27 – 28; independent claim 24) "a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies"

Independent claim 37

Independent claim 37	**(Page; Line)** – application as filed
A server, comprising:	(5; 25) "server 20" (6; 12 – 15) "The server 20 is a SyncML DM server. It issues SyncML DM commands to the mobile telephone 10 via the input/output interface 21 and correctly interprets responses from the mobile telephone 10" Figure 1
a memory configured to store code identifying first data and specifying execution of an unidentified executable on	(7; 15 to 8; 3) "The server 20 stores set-up code 50 in memory 23, which is usable with multiple devices without adaptation... The set-up code 50... comprises... a second portion 54 for carrying out executables." (3; 19 – 20; independent claim 1) "a command identifying

the first data; and	first data" (3; 27 – 28) "wherein the command identifies a first leaf node of a hierarchical nodular data structure" (4; 13; independent claim 19) "command identifying the first data" (4; 20 – 21; independent claim 22) "specifying execution of an unidentified executable on the first data" (4; 27 – 28; independent claim 24) "a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (10; 23 – 27) "The portion 54 for carrying out executables is processed as follows. The code is parsed to identify the first sub-portion. The first sub-portion is parsed to identify the URI specified by the element *LocURI* contained within the element *source.* The element LocURI identifies a leaf node within the newly created sub-tree of the management tree"
an interface configured to transmit the code.	(6; 11 – 15) "The server 20 comprises an input/output interface 21 connected to the cellular radio network 18 either directly or indirectly, a processor 22 and a memory 23. The server 20 is a SyncML DM server. It issues SyncML DM commands to the mobile telephone 10 via the input/output interface 21 and correctly interprets responses from the mobile telephone 10" (7; 15 – 18) "The server 20 stores set-up code 50 in memory 23, which is usable with multiple devices without adaptation. The server 20 in response to the download initiation message initiates a SyncML Data Management (DM) Session 62. The DM session 62 is used to transfer the stored set-up code 50 to the mobile device 10" (7; 25 – 28) "The set-up code may be sent by any suitable means such as a Short Message Service (SMS) message or, if the device is a personal digital assistant without mobile telephone capabilities via IR, Bluetooth or a serial data connection such as USB"

Independent claim 40

Independent claim 40	**(Page; Line)** – application as filed
A server, comprising:	(5; 25) "server 20" (6; 12 – 15) "The server 20 is a SyncML DM server. It issues SyncML DM commands to the mobile telephone 10 via the input/output interface 21 and correctly interprets responses from the mobile telephone 10" Figure 1
a memory configured to store commands,	(7; 15 to 8; 3) "The server 20 stores set-up code 50 in memory 23, which is usable with multiple devices without adaptation... The set-up code 50... comprises... a second portion 54 for carrying out executables."
execution of which resulting in creation at an electronic device, a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node, and	(4; 3 – 5; independent claim 18) "commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (4; 25 – 27; independent claim 24) "commands for creating at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (6; 23 – 30) "A management tree is a hierarchical nodular data structure by which the management client interacts with the mobile telephone 10. The MC may store or retrieve values from the tree and manipulate the properties of the tree. The management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child (dependent) nodes but can store a value..." (8; 1 – 2) "There is a first portion 52 for creating a management tree or updating an existing management tree" (8; 5 – 7) "The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree" Dependent claim 16 Figure 4
configured to store a further command	(8; 1 – 2) "There is a second portion 54 for carrying out executables."

identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first leaf node; and	(3; 19 – 20; independent claim 1) "a command ..." (3; 27 – 28) "wherein the command identifies a first leaf node of a hierarchical nodular data structure" (4; 20 – 21; independent claim 22) "specifying execution of an unidentified executable on the first data" (4; 27 – 28; independent claim 24) "a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies"
a transmitter configured to transmit the stored instructions	(6; 11 – 15) "The server 20 comprises an input/output interface 21 connected to the cellular radio network 18 either directly or indirectly, a processor 22 and a memory 23. The server 20 is a SyncML DM server. It issues SyncML DM commands to the mobile telephone 10 via the input/output interface 21 and correctly interprets responses from the mobile telephone 10" (7; 15 – 18) "The server 20 stores set-up code 50 in memory 23, which is usable with multiple devices without adaptation. The server 20 in response to the download initiation message initiates a SyncML Data Management (DM) Session 62. The DM session 62 is used to transfer the stored set-up code 50 to the mobile device 10" (7; 25 – 28) "The set-up code may be sent by any suitable means such as a Short Message Service (SMS) message or, if the device is a personal digital assistant without mobile telephone capabilities via IR, Bluetooth or a serial data connection such as USB"

Independent claim 41

Independent claim 41	**(Page; Line)** – application as filed

A computer program product comprising program instructions embodied on a tangible computer-readable medium, execution of the program instructions resulting in operations comprising:	(4; 3 – 6) "The operation of the processor 11 is controlled by software stored in the memory 13 and loaded into the processor. In operation, the processor receives and transmits data via the transceiver 12 and writes and reads data from the memory 13"
automatically determining, from metadata of first data, a content type of the first data;	(8; 15 – 17) "<Meta>... <Format>... <Type>" (10; 29 – 11; 1) "The DM client...reads the ... properties contained within *meta*. The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data..." Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12
automatically identifying an executable using the content type determined from the metadata; and	(3; 22 – 23; independent claim 1) "automatically identifying an executable from the determined property" (3; 29; independent claim 17) "identifying an executable from the determined property" (4; 7; independent claim 18) "identifying an executable from the determined property" (4; 14 – 15; independent claim 19) "means for identifying an executable from the determined property" (10; 12 – 15) "It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (11; 3 – 13) "The DM client associates possible *Formats* and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player..."
enabling the first data to be operated on using the identified executable.	(3; 22 – 23; independent claim 1) "operating on the identified first data using the identified executable" (3; 30 – 31; independent claim 17) "operating on data stored at the identified first leaf node using the identified executable" (4; 8 – 9; independent claim 18) "operating on the first data stored at the first leaf node using the identified executable" (4; 15 – 16; independent claim 19) "operating on the

	identified first data using the identified executable"

Independent claim 43

Independent claim 43	**(Page; Line)** – application as filed
A method, comprising:	Title (3; 18) (3; 25) (3; 33) Independent claim 1 Independent claim 17 Independent claim 18 Independent claim 30 Figures 1 to 4
receiving a first command at an electronic device, the first command specifying creation of a leaf node in a hierarchical data structure, and identifying first data to be stored at the leaf node and metadata indicating a content type of the first data;	(4; 3 – 5; independent claim 18) "commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (4; 25 – 27; independent claim 24) "commands for creating at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (6; 23 – 30) "A management tree is a hierarchical nodular data structure by which the management client interacts with the mobile telephone 10. The MC may store or retrieve values from the tree and manipulate the properties of the tree. The management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child (dependent) nodes but can store a value…" (8; 1 – 2) "There is a first portion 52 for creating a management tree or updating an existing management tree" (8; 5 – 7) "The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree" Dependent claim 16 Figure 4

	(8; 15 – 17) "<Meta>... <Format>... <Type>" (10; 29 – 11; 1) "The DM client...reads the ... properties contained within *meta*. The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data..." Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12
creating the leaf node at the electronic device;	(6; 32 – 33) "New nodes can be created and the values at certain leaf nodes can be changed" (8; 5 – 7) "The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree"
receiving a second command, at the electronic device, that specifies execution of an unidentified executable on the first data stored at the created leaf node;	(3; 19 – 20; independent claim 1) "a command ..." (3; 27 – 28) "wherein the command identifies a first leaf node of a hierarchical nodular data structure" (4; 20 – 21; independent claim 22) "specifying execution of an unidentified executable on the first data" (4; 27 – 28; independent claim 24) "a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies"
determining, from the metadata, the content type of the first data;	(8; 15 – 17) "<Meta>... <Format>... <Type>" (10; 29 – 11; 1) "The DM client...reads the ... properties contained within *meta*. The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data..." Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12
identifying an executable using the content type determined from the	(3; 22 – 23; independent claim 1) "automatically identifying an executable from the determined property" (3; 29; independent claim 17) "identifying an executable

metadata; and	from the determined property" (4; 7; independent claim 18) "identifying an executable from the determined property" (4; 14 – 15; independent claim 19) "means for identifying an executable from the determined property" (10; 12 – 15) "It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (11; 3 – 13) "The DM client associates possible *Formats* and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player…"
operating on the first data using the identified executable.	(3; 22 – 23; independent claim 1) "operating on the identified first data using the identified executable" (3; 30 – 31; independent claim 17) "operating on data stored at the identified first leaf node using the identified executable" (4; 8 – 9; independent claim 18) "operating on the first data stored at the first leaf node using the identified executable" (4; 15 – 16; independent claim 19) "operating on the identified first data using the identified executable

Independent claim 44

Independent claim 44	**(Page; Line)** – application as filed
An electronic device, comprising:	Title (1; 7) "Embodiments of the present invention relate to configuring an electronic device" (1; 30 – 31) "It is therefore desirable to provide some form of automatic set-up process for electronic devices and, in particular, mobile cellular telephones" (1; 33 – 2; 1) "It would be desirable for this set-up process to be such that it can be used with more than one electronic device without specific adaptation for that device" (3; 18 – 19) "According to one embodiment of the invention there is provided a method for automatically configuring an electronic device comprising"
a receiver configured to	(5; 24 – 28) "…a mobile cellular telephone, comprises… a

receive a first command at an electronic device, the first command specifying creation of a leaf node in a hierarchical data structure, and identifying first data to be stored at the leaf node and metadata indicating a content type of the first data; and	cellular radio transceiver 12…" (3; 19 – 20) "receiving at an electronic device a command identifying first data" (4; 13) "means for receiving a command identifying the first data" (4; 3 – 5; independent claim 18) "commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (4; 25 – 27; independent claim 24) "commands for creating at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node" (6; 23 – 30) "A management tree is a hierarchical nodular data structure by which the management client interacts with the mobile telephone 10. The MC may store or retrieve values from the tree and manipulate the properties of the tree. The management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child (dependent) nodes but can store a value…" (8; 1 – 2) "There is a first portion 52 for creating a management tree or updating an existing management tree" (8; 5 – 7) "The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree" Dependent claim 16 Figure 4 (8; 15 – 17) "<Meta>… <Format>… <Type>" (10; 29 – 11; 1) "The DM client…reads the … properties contained within *meta*. The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data…" Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12
a processor configured to create the leaf node at the	(5; 24 – 28) "…mobile cellular telephone, comprises: a processor 11…" (6; 17 – 21) "In the mobile telephone 10, processor 11

electronic device, wherein the receiver is further configured to receive a second command that specifies execution of an unidentified executable on the first data stored at the created leaf node, and the processor is further configured to determine, from the metadata, a content type of the first data, to identify an executable using the content type determined from the metadata, and to operate on the first data using the identified executable.	operates as a management client (MC) and can maintain a management tree data structure 100 in the memory 13. The MC correctly interprets SyncML DM commands received from the server, executes appropriate actions in the mobile telephone 10 and sends back relevant responses to the issuing management server via the transceiver 12" (10; 17 – 18) "The DM client of the mobile cellular telephone processes the received set-up code 50" (6; 32 – 33) "New nodes can be created and the values at certain leaf nodes can be changed" (8; 5 – 7) "The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree" (3; 19 – 20; independent claim 1) "a command ..." (3; 27 – 28) "wherein the command identifies a first leaf node of a hierarchical nodular data structure" (4; 20 – 21; independent claim 22) "specifying execution of an unidentified executable on the first data" (4; 27 – 28; independent claim 24) "a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node" (4; 30 – 31) "Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable" (10; 11 – 15) "This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (8; 15 – 17) "<Meta>... <Format>... <Type>" (10; 29 – 11; 1) "The DM client...reads the ... properties contained within *meta*. The DM client uses the content of the F*ormat* element and/or the content of the *Type* to identify the content type of the data..." Dependent claim 2 Dependent claim 10 Dependent claim 11 Dependent claim 12 (3; 22 – 23; independent claim 1) "automatically identifying

	an executable from the determined property" (3; 29; independent claim 17) "identifying an executable from the determined property" (4; 7; independent claim 18) "identifying an executable from the determined property" (4; 14 – 15; independent claim 19) "means for identifying an executable from the determined property" (10; 12 – 15) "It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies" (11; 3 – 13) "The DM client associates possible *Formats* and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player..." (3; 22 – 23; independent claim 1) "operating on the identified first data using the identified executable" (3; 30 – 31; independent claim 17) "operating on data stored at the identified first leaf node using the identified executable" (4; 8 – 9; independent claim 18) "operating on the first data stored at the first leaf node using the identified executable" (4; 15 – 16; independent claim 19) "operating on the identified first data using the identified executable"

(6) GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. The first grounds for rejection presented for review by the Board is whether claims 1, 3-10, 12-21, 34, 41 and 43-44 are obvious under 35 USC Section 103(a) over Rao et al. (US Patent 6,978,453, hereinafter "Rao") in view of SynchML Meta-Information DTD, hereinafter "DTD" and Szeto (US Patent 7,188,143, hereinafter "Szeto").

B. The second issue presented for review by the Board is whether claims 22-28 and 35-40 are obvious over Rao in view of Szeto.

(7) ARGUMENT

Appellant respectfully submits that the claims are not obvious in view of the afore-cited references.

A. (CLAIMS 1-3-10, 12-21, 34, 41 and 43-44)

By way of background, in the Final Office Action, claims 1-21, 34, 41 and 43-44 were rejected under 35 USC Section 103(a) as being obvious over Rao in view of DTD and Szeto. *See pages 6-25 of the Final Office Action.* As claims 2 and 11 were previously canceled, as noted above, the below analysis addresses claims 1, 3-10, 12-21, 34, 41 and 43-44 as being finally rejected in view of Rao, DTD and Szeto.

Of claims 1, 3-10, 12-21, 34, 41 and 43-44, claims 1, 17, 18, 19, 34, 41, 43 and 44 are independent. Claims 3-10 and 12-16 depend ultimately from claim 1. Claim 20 depends from claim 19. Claim 21 depends from claim 20.

B. (CLAIMS 22-28 and 35-40)

By way of background, in the Final Office Action, claims 22-28 and 35-40 were rejected under 35 U.S.C. §103(a) as being unpatentable over Rao in view of Szeto. *See pages 26-34 of the Final Office Action.*

Of claims 22-28 and 35-40, claims 22, 24, 35, 36, 37 and 40 are independent. Claims 23 and 25-28 depend from claim 22. Claims 38-39 depend from claim 37.

Appellant respectfully disagrees with the Examiner's rejection of all of the claims and presents the below analysis in support of the patentability of the claims. Reconsideration and withdrawal of all rejections set forth in the afore-referenced Final Office Action is respectfully requested.

ANALYSIS

Rejection under 35 U.S.C. 103(a) over US Patent 6,978,453 (Rao) in view of "SyncML Meta-Information DTD" and US Patent 7,188,143 (Szeto)

Claims 1 and 3 to 10 and 12 to 16

Independent claim 1 recites:

A) A method comprising:
receiving at an electronic device a command specifying execution of an unidentified executable on first data;

B) automatically determining, from metadata of the first data, a content type of the first data;

C) automatically identifying an executable using the content type determined from the metadata; and

D) operating on the first data using the identified executable.

It is respectfully pointed out that letter notation and emphasis to the claims are provided herein for ease of reference in understanding Appellant's analysis.

Rao

Rao discloses a system for employing SyncML DM for updating firmware in mobile handsets and other devices (abstract). Rao refers to SyncML "enhancement commands, such as, for example, GetFirmwareUpdate, VerifyFirmwareUpdate, SaveFirmwareUpdate, ApplyFirmwareUpdate [and] ConfirmFirmwareUpdate" (column 8, lines 16 to 23).

Rao indicates, at column 8, lines 25 to 28, that "the SyncML DM protocol allows the enhancement commands to be executed on nodes of [a] management tree in [a] mobile device 107".

According to Rao, a "SyncML management server may employ the exec command to invoke the enhancement commands associated with the firmware updates in the mobile handset. The exec command may launch a process that initiates the firmware update download in accordance with the parameters provided in the exec command" (column 8, lines 35 to 42).

Appellant agrees with the Examiner that Rao does not disclose A because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec

commands to invoke enhancement commands for the firmware update).

Appellant also agrees with the Examiner that Rao does not disclose B because, for example, there is no mention of determining a content type from metadata whatsoever. Furthermore, the system used in Rao is for a firmware update and is therefore specific to taking a target firmware update management object and updating it (column 11, lines 21 to 22; column 12, lines 13 to 15). Therefore, the update process is essentially the transfer of known data and Rao does not need to know the content type of the data.

Appellant also agrees with the Examiner that Rao does not disclose C because, for example, there is no mention of determining a content type from metadata whatsoever, as explained above. Furthermore, in Rao, the execution is of an update process and there is no disclosure of what happens to the firmware update data once it has been downloaded and applied (i.e. Rao merely facilitates the transfer of firmware update data and does not discuss the application of firmware). Therefore, Rao uses a system to execute an update process and does not discuss the identification of particular executables for operation on particular content types. Rao does not discuss using metadata from a firmware update, whether for determining an executable or for any other reason. Instead, Rao merely specifies explicit commands to fetch and apply firmware updates.

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because, for example, Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Therefore, Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of DTD and Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of

DTD and/or Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

DTD

More particularly, the DTD reference is a Document Type Definition specification that defines a set of mark-up that may be used to identify meta-information associated with a SyncML command, data item or collection (page 5, section 1). The document gives examples of how the element type in SyncML can be used to specify the content information in the data element. In particular, the encoding format and the media type of the content information in the data element may be specified (page 6, section 5.2; page 11, section 5.10). Page 11, under the heading "Restrictions", states that the content type should be a registered MIME (Multipurpose Internet Mail Extension) content-type or a URN (Uniform Resource Name).

DTD does not disclose (or suggest) A because there is no method disclosed that uses any commands or execution whatsoever.

DTD does not disclose (or suggest) C because there is no disclosure of any executables whatsoever.

DTD does not disclose (or suggest) D because there is no execution or executable disclosed whatsoever.

DTD contains information about SyncML, but does not describe any implementation or uses of SyncML technology. Thus, DTD does not explain how content information in a data element can be used to produce any technical effect.

DTD therefore does not contain any teaching, suggestion or motivation to adapt the information it contains to arrive at the claimed invention. It is respectfully asserted that the TSM test (teaching, suggestion and motivation) provides helpful insights into the non-

obviousness of the subject claims. Moreover, pursuant to *KSR*, there is no reason to modify the teachings of DTD in an attempt to arrive at the subject claims.

Szeto

Regarding the Szeto reference cited by the Examiner, Szeto relates to techniques for controlling an application in an instant messaging (IM) environment (abstract). Szeto indicates that an "instant messaging environment is a shared environment which exists between 2 or more instant messaging users" (column 4, lines 61 to 63) [i.e. between a sending client and a receiving client]. The "current environment affects how user interface commands sent from instant messaging client 212 [receiving client] to conversation user interface 216 [of the receiving client] are processed" (column 6, lines 28 to 30).

Examples of different types of environment are provided in Figs 9A, 9B and 9C. Fig. 9A illustrates a "cartoon messaging environment", Fig. 9B illustrates a "snow theme environment" and Fig. 9C illustrates a "stock ticker environment". These different environments redefine functions so that received messages are handled differently, dependent on the current environment being used. For example, in the cartoon messaging environment, the environment defines functions such that when a new message is sent, an existing text bubble displayed above a sender's cartoon character is deleted and replaced by a new text bubble comprising the new message, whereas in a generic environment, the new message would simply be added to a history window and the existing messages would not be deleted (column 5, lines 42 to 44; column 11, lines 30 to 34).

Szeto discloses at column 3, lines 11 to 19 that in "one embodiment, a method is disclosed for selecting, at a first client [i.e. sending client], the application [e.g. movie trailer] in the instant messaging environment, configuring an instant messaging control message for the application, including an identifier related to the application [movie trailer] selected at the first client [sending client], sending the instant messaging control message to a second client [i.e. receiving client], and executing the application in the instant messaging environment using the control message to retrieve the application from a server, unless the application has

been previously called by the user."

The portion of Szeto that the Examiner appears to consider particularly relevant is column 12, line 66 to column 13, line 16:

> "In step 1202, IM client 202 [receiving client] (FIG. 2 or 10) evaluates an IM message [received from the sending client]. From the [identifier of the] IM message, the IM client 202 [receiving client] determines the application type (i.e. movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204. Using an identifier, the IM application is retrieved in step 1206. In step 1208, a decision is made as to whether a supporting application is required such as a media player (Real Player, Windows Media Player), content viewer (Adobe Illustrator, Reader, etc.), or other media-based display application. If required, the supporting application is launched in step 1210".

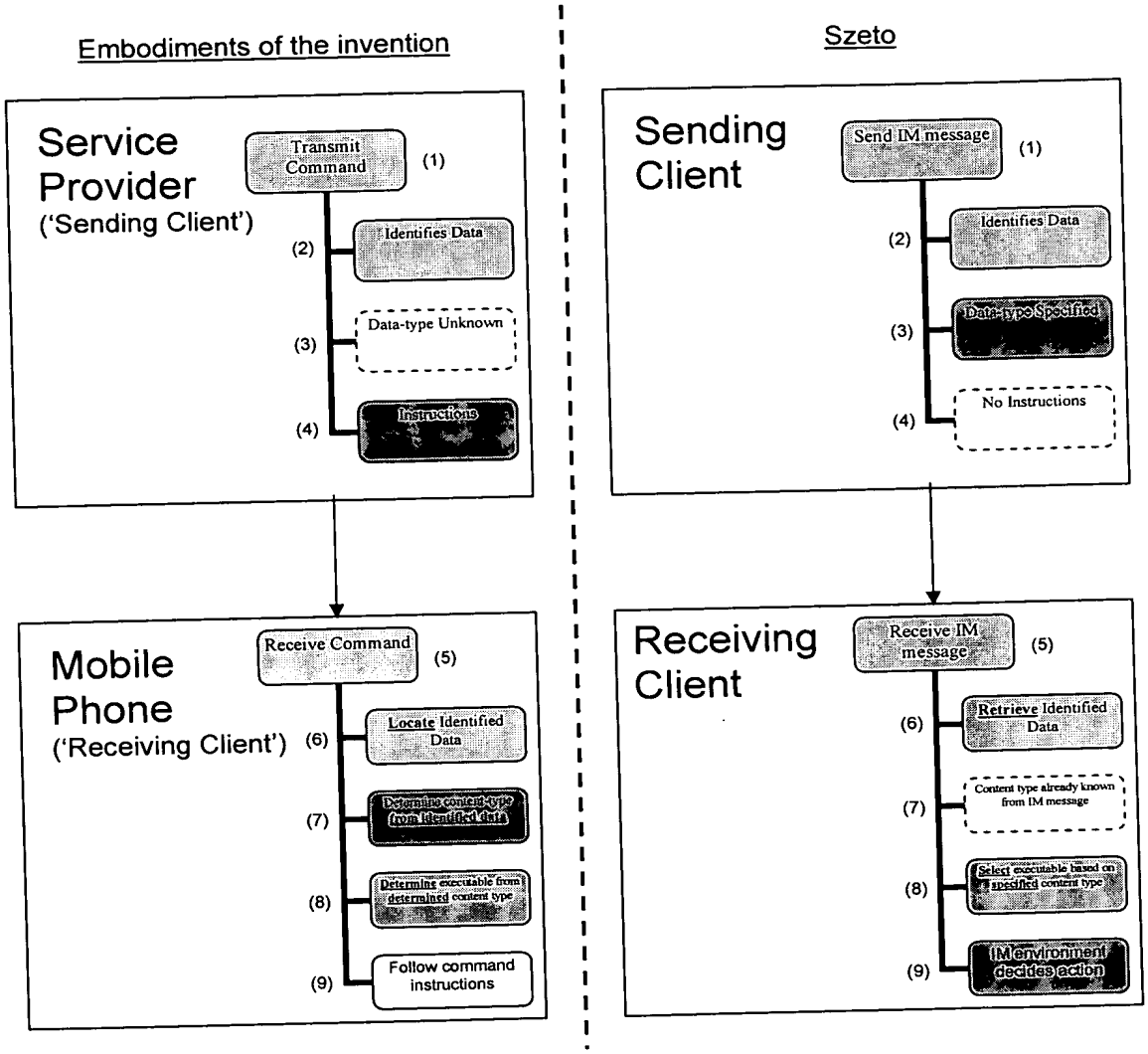
Column 13, lines 17 to 25, gives an example of a movie trailer (IM application) being retrieved from a movie server using a movie trailer identifier.

Appellant further respectfully notes the following:

> The "IM application" in Szeto is the actual data to be executed and not the executable itself (i.e. conventionally we would call a media player the application and the movie trailer the data to be executed by the application, whereas in Szeto, the movie trailer is known as the IM application which is executed by a "supporting application" such as a media player) (column 12, lines 58 to 59; column 13, lines 7 to 10). Therefore the "supporting application" in Szeto is the executable.

In the following diagram, "service provider" and "mobile phone" have been used by way of example only and it is to be understood that the use of these words does not limit the scope of any of the claims in any way. It is presented merely for illustrative purposes of an example.

Summary of Szeto vs. embodiments of the invention



Appellant respectfully disagrees with the Examiner that Szeto discloses A.

Szeto does not disclose or suggest 'A' because, for example:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the claimed invention, the execution is commanded by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the

execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.

- column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

Contrastingly, embodiments of the claimed invention use commands sent by a service provider (i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the claimed invention, the executable is determined at the receiving client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the claimed invention, the receiving client determines a suitable executable to be used by reading the properties of the metadata of the first data at the receiving client (page 10, line 29 to page 11, line 1).

Szeto does not disclose or suggest 'B' because:

a) In agreement with the Examiner, there is no disclosure whatsoever of metadata.

b) **In Szeto, the "content type" is specified at the sending client, whereas in embodiments of the claimed invention, the content type is determined at the receiving client.** As illustrated in the above diagram at points 3 and 7, the application

type (content type) of the data (to be retrieved) in Szeto is specified by the sending client, using an identifier, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client"
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Clearly, in Szeto, the identifier within the IM message provides information about the content type of the IM application and also the location from where the IM application should be retrieved, i.e. the identifier may be a movie trailer identifier that: (i) identifies the IM application to be retrieved as a movie trailer (content type); and (ii) specifies the movie trailer to be retrieved from a movie server (location for retrieval). Therefore the "content type" of the IM application is specified at the sending client. Contrastingly, embodiments of the invention use, for example, a DM client of a mobile phone (receiving client) to determine the content type of the identified first data from metadata of the first data (page 10, line 29 to page 11, line 1).

c) **In Szeto, the content type is identified (by the sending client) using an identifier, whereas in the present invention, the content type is determined (by the receiving client) from "first data".** As illustrated in the above diagram at point 6, the "first data" (i.e. the data to be executed) in Szeto, requires retrieval from a server and is not present in the IM message. Therefore, as illustrated at point 7 in Szeto, because the content type is determined from the IM message (which does not contain the data to be executed), it cannot be determined from first data.

- column 3, lines 11 to 19: "using the control message [at the receiving client] to retrieve the application from a server"
- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application

identifier [received in the IM message sent by the sending client], to execute the instant messaging application".

Contrastingly, embodiments of the claimed invention use, for example a DM client of a mobile phone (receiving client) to access the identified data in order to determine the content type of the identified data (page 10, line 29 to page 11, line 1).

Szeto does not disclose or suggest 'C' because:

a) As above, there is no disclosure of metadata.

b) As above, the content type is specified by the sending client and not determined at the receiving client.

c) As above, the content type is specified using an identifier and is not determined from first data.

d) **In Szeto, the receiving client is told the content type, and hence the executable required, by the sending client**. As illustrated in the above diagram at points 7 and 8, the IM environment at the receiving client uses the identifier in the IM message to select a supporting application.

- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [in the IM message sent by the sending client], to execute the instant messaging application";
- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Contrastingly, embodiments of the claimed invention determine a suitable executable

to be used by reading the properties of the metadata of the first data at the receiving client (page 10, line 29 to page 11, line 1).

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to specify execution in one or more mobile phone's using a non-specific command message. Therefore the control aspect of embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the Appellant's embodiments of the invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the claimed invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

None of Rao, DTD and Szeto individually disclose or suggest, for example, the features A and C of the claimed invention. Therefore, Rao, DTD and Szeto, when combined cannot disclose or suggest the features A and C of the claimed invention. Nor is there any reason to combine and modify the teachings of the cited references in an attempt to arrive at the subject claims.

Alleged Combination of Rao + DTD

The Examiner, in the Advisory Action, comments that Rao is related to SyncML technology and therefore there is motivation to combine it with the SyncML operational specification.

Although the Appellant agrees with the Examiner that DTD may be in a similar field of technology to Rao with regard to the use of SyncML, the Appellant disagrees that these documents can be combined in an attempt to arrive at the claimed invention or indeed to disclose or suggest, for example, features A or C.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. DTD is an information document that discloses a set of mark-up used by SyncML to identify meta-information associated with a SyncML command or data item or collection. Rao specifies named executables to be used when applying a firmware update. DTD provides information only and no implementation/use information. Therefore, the content information determined from metadata (as disclosed in DTD) is not required for the firmware update in Rao. There is no reason why one skilled in the art would use content type determined from metadata when applying a firmware update, particularly when Rao uses named executables.

Therefore, the combination of these documents does not disclose or suggest the claimed invention. The combination does not disclose any feature of the claims that the documents do not disclose individually. Furthermore, there is no motivation to combine the documents because Rao teaches the use of named executables and there is no requirement to know

content type of the firmware update.

Alleged Combination of Rao + Szeto

Szeto relates to controlling an application in a shared environment which exists between two or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, it is incomprehensible why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Furthermore, the combination of these documents does not disclose any feature of A-C of independent claim 1.

Alleged Combination of Szeto + DTD

DTD is only relevant for SyncML technology, and therefore there is no reason why one skilled in the art would combine its teachings with Szeto. It is unclear why one skilled in the art would use SyncML technology in IM messaging.

The Appellant therefore disagrees with the permissibility of combining Szeto and DTD. It is not known or specified from either document to use an instant messaging application, or

indeed an instant message with SyncML technology. Furthermore, as Szeto specifies the content type of the IM application at the sending client, it is unclear why "determining content type from metadata of first data" would be of use.

There is therefore no teaching, suggestion or motivation in Szeto to combine it with DTD whatsoever. Nor is there any reason to do so.

Alleged Combination of Rao + DTD + Szeto

There would thus be no reason to combine Rao and DTD with Szeto because, for example, Szeto is not relevant to SyncML technology and Rao and DTD are specific to SyncML technology.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is concerned with firmware updating, and DTD is a Document Type Definition specification for SyncML. The object to be attained in Rao is how to update firmware using SyncML. The object to be attained in DTD is how to use SyncML element types. Szeto does not relate to firmware updates, nor does it relate to using element types in SyncML.

Thus, one of ordinary skill in the art would not look to combine the teachings of Rao or DTD with Szeto. Nor would the combination result in the claimed invention.

Moreover, the Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata". Szeto does not identify "the corresponding application" (supporting application) using metadata (as disclosed in DTD) but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore, the use of Szeto in combination with Rao and/or DTD would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific

command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified data, whereas in the present invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the claimed invention determines how to do it. Thus, Szeto teaches away from embodiments of the invention by specifying the control of received messages at the receiving client and not at the sending client.

Accordingly, for at least the above reasons, independent claim 1 is considered to be new and non-obvious in view of the cited art. Similarly, claims 3-10 and 12-16, although containing their own allowable subject matter, also are considered to be new and non-obvious at least in view of their dependency from an allowable independent claim.

Claim 17

Independent claim 17 currently recites:

E. A method, comprising:
transferring code comprising a command to an electronic device, wherein the command specifies execution of an unidentified executable on first data stored at a first leaf node of a hierarchical nodular data structure;

F. determining, from metadata of the first leaf node, a content type of the first data;

G. identifying an executable using the content type determined from the metadata of the identified first leaf node; and

H. operating on the first data, stored at the identified first leaf node using the identified executable.

Rao

Appellant agrees with the Examiner that Rao does not disclose E because, for example, Rao

uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

Appellant also agrees with the Examiner that Rao does not disclose F because, for example, there is no mention determining a content type from metadata whatsoever. Furthermore, the system used in Rao is for a firmware update and is therefore specific to taking a target firmware update management object and updating it (column 11, lines 21 to 22; column 12, lines 13 to 15). Therefore, the update process is essentially the transfer of known data and Rao does not need to know the content type of the data.

Appellant agrees with Examiner that Rao does not disclose G because, for example, there is no mention determining a content type from of metadata whatsoever, as explained above. Furthermore, in Rao, the execution is of an update process and there is no disclosure of what happens to the firmware update data once it has been downloaded and applied (i.e. Rao merely facilitates the transfer of firmware update data and does not discuss the application of firmware). Therefore, Rao uses a system to execute an update process and does not discuss the identification of particular executables for operation on particular content types. Rao does not discuss using metadata from a firmware update, whether for determining an executable or for any other reason. Instead, Rao merely specifies explicit commands to fetch and apply firmware updates.

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of DTD and Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of DTD and/or Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

DTD

More particularly, DTD does not disclose (or suggest) E because there is no method disclosed that uses any commands or execution whatsoever.

DTD does not disclose (or suggest) G because there is no disclosure of any executables whatsoever.

DTD does not disclose (or suggest) H because there is no execution or executable disclosed whatsoever.

DTD contains information about SyncML but does not discuss any implementation or uses of SyncML technology. Thus, DTD does not explain how content information in a data element can be used to produce any technical effect.

DTD therefore does not contain any teaching, suggestion or motivation to adapt the information it contains to arrive at the claimed invention. It is respectfully asserted that the TSM test (teaching, suggestion and motivation) provides helpful insights into the non-obviousness of the claims. Moreover, pursuant to *KSR*, there is no reason to modify the teachings of DTD in an attempt to arrive at the subject claims.

Szeto

Appellant respectfully disagrees with the Examiner that Szeto discloses E.

Szeto does not disclose or suggest 'E' because, for example:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the claimed invention, the execution is commanded by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.

- column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

Contrastingly, embodiments of the claimed invention use commands transferred from a service provider (i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the claimed invention, the executable is determined at the receiving client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the claimed invention, the executable is "unidentified" at the sending client and the receiving client determines a suitable executable to be used by reading the properties of the metadata of the first data at the receiving client (page 10, line 29 to page 11, line 1).

c) There is no disclosure whatsoever in Szeto of a first leaf node or a hierarchical nodular data structure.

Additionally, Szeto does not disclose or suggest 'F' because:

a) In agreement with the Examiner, there is no disclosure whatsoever of metadata.

b) **In Szeto, the "content type" is specified at the sending client, whereas in embodiments of the invention invention, the content type is determined at the receiving client**. As illustrated in the above diagram at points 3 and 7, the application type (content type) of the data (to be retrieved) in Szeto is specified by the sending client, using an identifier, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client"
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Clearly, in Szeto, the identifier within the IM message provides information about the content type of the IM application and also the location from where the IM application should be retrieved, i.e. the identifier may be a movie trailer identifier that: (i) identifies the IM application to be retrieved as a movie trailer (content type); and (ii) specifies the movie trailer to be retrieved from a movie server (location for retrieval). Therefore the "content type" of the IM application is specified at the sending client. Contrastingly, embodiments of the invention use, for example, a DM client of a mobile phone (receiving client) to determine the content type of the identified first data from metadata of the first leaf node (page 10, line 29 to page 11, line 1).

c) **In Szeto, the content type is identified (by the sending client) using an identifier, whereas in embodiments of the invention, the content type is determined (by the receiving client) from "metadata of the identified first leaf node".** As illustrated in the above diagram at point 6, the "first data" (i.e. the data to be executed) in Szeto,

requires retrieval from a server and is not present in the IM message. Therefore, as illustrated at point 7 in Szeto, because the content type is determined from the IM message (which does not contain the data to be executed), it cannot be determined from metadata of the first leaf node (which stores the first data), i.e.

- column 3, lines 11 to 19: "using the control message [at the receiving client] to retrieve the application from a server"
- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application".

Contrastingly, embodiments of the invention use, for example a DM client of a mobile phone (receiving client) to access the identified data in order to determine the content type of the identified data (page 10, line 29 to page 11, line 1).

d) There is no disclosure whatsoever in Szeto of a first leaf node.

Szeto also does not disclose or suggest 'G' because:

a) As above, there is no disclosure of metadata.
b) As above, the content type is specified by the sending client and not determined at the receiving client.
c) As above, the content type is specified using an identifier and is not determined from metadata of the first leaf node.
d) **In Szeto, the receiving client is told the content type, and hence the executable required, by the sending client**. As illustrated in the above diagram at points 7 and 8, the IM environment at the receiving client uses the identifier in the IM message to select a supporting application.
    - Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [in the IM message sent by the sending client], to execute the instant messaging application";

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Contrastingly, embodiments of the invention determine a suitable executable to be used by reading the properties of the metadata of the first leaf node at the receiving client (page 10, line 29 to page 11, line 1).

e) There is no disclosure whatsoever in Szeto of a first leaf node.

Szeto further does not disclose 'H' because there is no disclosure whatsoever of a first leaf node or first data stored at the identified first leaf node.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant

messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to specify execution in one or more mobile phone's using a non-specific command message. Therefore the control aspect of embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from Appellant's embodiments of the invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the claimed invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

None of Rao, DTD and Szeto individually disclose or suggest, for example, the features E and G of the claimed invention. Therefore Rao, DTD and Szeto, when combined cannot disclose or suggest the features E and G of the claimed invention. Nor is there any reason to combine and modify the teachings of these references in an attempt to arrive at the subject claims.

Alleged Combination of Rao + DTD

The Examiner, in his Advisory Action, comments that Rao is related to SyncML technology and therefore there is motivation to combine it with the SyncML operational specification.

Although the Appellant agrees with the Examiner that DTD may be in a similar field of technology to Rao with regard to the use of SyncML, the Appellant disagrees that these documents can be combined in an attempt to arrive at the claimed invention or indeed to disclose any additional feature of the claims that DTD or Rao, when taken individually, does not disclose.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. DTD is an information document that discloses a set of mark-up used by SyncML to identify meta-information associated with a SyncML command or data item or collection. Rao specifies named executables to be used when applying a firmware update. DTD provides information only and no implementation/use information. Therefore, the content information determined from metadata (as disclosed in DTD) is not required for the firmware update in Rao. There is no reason why one skilled in the art would use content type determined from metadata when applying a firmware update, particularly when Rao uses named executables.

Therefore, the combination of these documents does not disclose or suggest the claimed invention. The combination does not disclose or suggest any feature of the claims that the documents do not disclose individually. Furthermore, there is no motivation to combine the documents because Rao teaches the use of named executables and there is no requirement to know content type of the firmware update.

Alleged Combination of Rao + Szeto

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific

"enhanced" commands. Therefore, there is no reason why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Furthermore, the combination of these documents does not disclose or suggest any feature of E-G of independent claim 17.

Alleged Combination of Szeto + DTD

DTD is only relevant for SyncML technology, and therefore there is no reason why one skilled in the art would combine its teachings with Szeto. It is unclear why one skilled in the art would use SyncML technology in IM messaging.

The Appellant therefore disagrees with the permissibility of combining Szeto and DTD. It is not known or specified from either document to use an instant messaging application, or indeed an instant message with SyncML technology. Furthermore, as Szeto specifies the content type of the IM application at the sending client, it is unclear why "determining content type from metadata of first data" would be of use.

There is therefore no teaching, suggestion or motivation in Szeto to combine it with DTD whatsoever. Nor is there any reason to do so.

Alleged Combination of Rao + DTD + Szeto

There would thus be no reason to combine Rao and DTD with Szeto because, for example, Szeto is not relevant to SyncML technology and Rao and DTD are specific to SyncML technology.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is concerned with firmware updating, and DTD is a Document Type Definition specification for SyncML. The object to be attained in Rao is how to update firmware using SyncML. The object to be attained in DTD is how to use SyncML element types. Szeto does not relate to firmware updates, nor does it relate to using element types in SyncML.

Thus, the combination of Rao and/or DTD with Szeto is considered impermissible, and is therefore non-obvious. One skilled in the art would not look to combine the teachings of these references. Nor would the combination disclose or suggest the claimed invention.

The Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata". Szeto does not identify "the corresponding application" (supporting application) using metadata (as disclosed in DTD) but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore the use of Szeto in combination with Rao and/or DTD would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified data, whereas in embodiments of the invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the claimed invention determines how to do it. Thus, Szeto teaches away from the claimed invention by specifying the control of received messages at the receiving client and not at the sending client.

Accordingly, for at least the above reasons, independent claim 17 is considered to be new and non-obviousness in view of the cited art.

Claim 18

Independent claim 18 currently recites:

I) A method, comprising:
receiving re-usable code at an electronic device wherein the code comprises:

J) commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and

K) a further command specifying execution of an unidentified executable on the first data stored at the first leaf node;

L) determining, from metadata stored at the first leaf node, a content type of the first data stored at the first leaf node;

M) identifying an executable using the content type determined from the metadata stored at the first leaf node; and

N) operating on the first data stored at the first leaf node using the identified executable.

Rao

Appellant agrees with the Examiner that Rao does not disclose K because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

Appellant agrees with the Examiner that Rao does not disclose L because, for example, there is no mention of determining a content type from metadata whatsoever. Furthermore, the system used in Rao is for a firmware update and is therefore specific to taking a target firmware update management object and updating it (column 11, lines 21 to 22; column 12, lines 13 to 15). Therefore, the update process is essentially the transfer of known data and Rao does not need to know the content type of the data.

Appellant agrees with the Examiner that Rao does not disclose M because, for example, there is no mention of determining a content type from metadata whatsoever, as explained above. Furthermore, in Rao, the execution is of an update process and there is no disclosure of what happens to the firmware update data once it has been downloaded and applied (i.e. Rao

merely facilitates the transfer of firmware update data and does not discuss the application of firmware). Therefore Rao uses a system to execute an update process and does not discuss the identification of particular executables for operation on particular content types. Rao does not discuss using metadata from a firmware update, whether for determining an executable or for any other reason. Instead, Rao merely specifies explicit commands to fetch and apply firmware updates.

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of DTD and Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of DTD and/or Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

DTD

DTD does not disclose (or suggest) K because there is no method disclosed that uses any commands or execution whatsoever.

DTD does not disclose (or suggest) M because there is no disclosure of any executables whatsoever.

DTD does not disclose (or suggest) N because there is no execution or executable disclosed whatsoever.

DTD does not disclose (or suggest) I because there is no disclosure of a re-usable code at an electronic device.

DTD does not disclose (or suggest) J because there is no disclosure of any commands nor is there any disclosure of creation of a hierarchical nodular data structure.

DTD contains information about SyncML but does not discuss any implementation or uses of SyncML technology. Thus, DTD does not explain how content information in a data element can be used to produce any technical effect.

DTD therefore does not contain any teaching, suggestion or motivation to adapt the information it contains to arrive at the claimed invention. It is respectfully asserted that the TSM test (teaching, suggestion and motivation) provides helpful insights into the non-obviousness of the claims. Moreover, pursuant to *KSR*, there is no reason to modify the teachings of DTD in an attempt to arrive at the subject claims.

Szeto

Appellant respectfully disagrees with the Examiner that Szeto discloses K.

Szeto does not disclose or suggest 'K' because, for example:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the claimed invention, the execution is commanded by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.
   - column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

Contrastingly, embodiments of the present invention use commands sent by a service provider (i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

b) **In Szeto, the executable is identified by the sending client, whereas in the present invention, the executable is determined at the receiving client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the claimed invention, the receiving client determines a suitable executable to be used by reading the properties of the metadata stored at the first leaf node at the receiving client (page 10, line 29 to page 11, line 1).

c) There is no disclosure whatsoever in Szeto of a first leaf node or first data stored at the first leaf node.

Szeto does not disclose or suggest 'L' because:

a) In agreement with the Examiner, there is no disclosure whatsoever of metadata.

b) **In Szeto, the "content type" is specified at the sending client, whereas in the present invention, the content type is determined at the receiving client**. As illustrated in the above diagram at points 3 and 7, the application type (content type) of the data (to be retrieved) in Szeto is specified by the sending client, using an identifier, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client"
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Clearly, in Szeto, the identifier within the IM message provides information about the content type of the IM application and also the location from where the IM application should be retrieved, i.e. the identifier may be a movie trailer identifier that: (i) identifies the IM application to be retrieved as a movie trailer (content type); and (ii) specifies the movie trailer to be retrieved from a movie server (location for retrieval). Therefore the "content type" of the IM application is specified at the sending client. Contrastingly, embodiments of the claimed invention use, for example, a DM client of a mobile phone (receiving client) to determine the content type of the identified first data from metadata stored at the first leaf node (page 10, line 29 to page 11, line 1).

c) **In Szeto, the content type is identified (by the sending client) using an identifier, whereas in the present invention, the content type is determined (by the receiving client) from "metadata stored at the first leaf node".** As illustrated in the above diagram at point 6, the "first data" (i.e. the data to be executed) in Szeto, requires retrieval from a server and is not present in the IM message. Therefore, as illustrated at point 7 in Szeto, because the content type is determined from the IM message (which does not contain the data to be executed), it cannot be determined from metadata stored at the first leaf node, i.e.

- column 3, lines 11 to 19: "using the control message [at the receiving client] to retrieve the application from a server"
- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application

identifier [received in the IM message sent by the sending client], to execute the instant messaging application".

Contrastingly, embodiments of the claimed invention use, for example a DM client of a mobile phone (receiving client) to access the identified data in order to determine the content type of the identified data (page 10, line 29 to page 11, line 1).

d) There is no disclosure whatsoever in Szeto of a first leaf node.

Szeto does not disclose or suggest 'M' because:

a) As above, there is no disclosure of metadata.

b) As above, the content type is specified by the sending client and not determined at the receiving client.

c) As above, the content type is specified using an identifier and is not determined from metadata stored at the first leaf node.

d) **In Szeto, the receiving client is told the content type, and hence the executable required, by the sending client**. As illustrated in the above diagram at points 7 and 8, the IM environment at the receiving client uses the identifier in the IM message to select a supporting application, i.e.

- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [in the IM message sent by the sending client], to execute the instant messaging application";
- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Contrastingly, embodiments of the claimed invention determine a suitable executable to be used by reading the properties of the metadata stored at the first leaf node at the receiving client (page 10, line 29 to page 11, line 1).

e) There is no disclosure whatsoever in Szeto of a first leaf node.

Szeto does not disclose or suggest 'N' because there is no disclosure whatsoever of a first leaf node nor is there any disclosure of first data stored at the identified first leaf node.

Szeto does not disclose or suggest I because there is no disclosure whatsoever of a re-usable code.

Szeto does not disclose or suggest J because there is no disclosure whatsoever of a hierarchical nodular data structure or of commands for the creation of a hierarchical nodular data structure.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the

receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to create a hierarchical nodular data structure at one or more mobile phones and specify execution using a non-specific command message. Therefore the control aspect of the present invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from embodiments of the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

None of Rao, DTD and Szeto individually disclose or suggest the features K and M of the claimed invention. Rao, DTD and Szeto, when combined cannot disclose or suggest the features K and M of the claimed invention.

Alleged Combination of Rao + DTD

The Examiner, in his Advisory Action, comments that Rao is related to SyncML technology and therefore there is motivation to combine it with the SyncML operational specification.

Although the Appellant agrees with the Examiner that DTD may be in a similar field of technology to Rao with regard to the use of SyncML, the Appellant disagrees that these documents can be combined in an attempt to arrive at the claimed invention or indeed to disclose any additional feature of the claims that DTD or Rao, when taken individually, does not disclose.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. DTD is an information document that discloses a set of mark-up used by SyncML to identify meta-information associated with a SyncML command or data item or

collection. Rao specifies named executables to be used when applying a firmware update. DTD provides information only and no implementation/use information. Therefore, the content information determined from metadata (as disclosed in DTD) is not required for the firmware update in Rao. There is no reason why one skilled in the art would use content type determined from metadata when applying a firmware update, particularly when Rao uses named executables.

Therefore, the combination of these documents does not disclose or suggest the claimed invention. The combination does not disclose any feature of the claims that the documents do not disclose individually. Furthermore, there is no motivation to combine the documents because Rao teaches the use of named executables and there is no requirement to know content type of the firmware update.

Alleged Combination of Rao + Szeto

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, it is incomprehensible why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Furthermore, the combination of these documents does not disclose any feature of I-M of independent claim 18.

Alleged Combination of Szeto + DTD

DTD is only relevant for SyncML technology, and therefore it is incomprehensible why the Examiner has combined this document with Szeto. It is unclear why one skilled in the art would use SyncML technology in IM messaging.

The Appellant therefore disagrees with the permissibility of combining Szeto and DTD. It is not known or specified from either document to use an instant messaging application, or indeed an instant message with SyncML technology. Furthermore, as Szeto specifies the content type of the IM application at the sending client, it is unclear why "determining content type from metadata of first data" would be of use.

There is therefore no teaching, suggestion or motivation in Szeto to combine it with DTD whatsoever. Nor is there any reason to do so.

Alleged Combination of Rao + DTD + Szeto

The combination Rao and DTD with Szeto is not obvious because, for example, Szeto is not relevant to SyncML technology and Rao and DTD are specific to SyncML technology.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is concerned with firmware updating, and DTD is a Document Type Definition specification for SyncML. The object to be attained in Rao is how to update firmware using SyncML. The object to be attained in DTD is how to use SyncML element types. Szeto does not relate to firmware updates, nor does it relate to using element types in SyncML.

Thus, the combination of Rao and/or DTD with Szeto is considered impermissible, and is therefore non-obvious. One skilled in the art would not look to combine the teachings of these references. Nor would the combination disclose or suggest the claimed invention.

The Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata". Szeto does not identify "the corresponding application" (supporting application) using metadata (as disclosed in DTD) but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore the use of Szeto in combination with Rao and/or DTD would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified data, whereas in the present invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the claimed invention determines how to do it. Thus, Szeto teaches away from embodiments of the claimed invention by specifying the control of received messages at the receiving client and not at the sending client.

Accordingly, for at least the foregoing reasons, independent claim 18 is considered to be new and non-obvious in view of the cited art.

Claims 19 to 21

Independent claim 19 currently recites:

O) An electronic device, comprising:
a memory configured to store first data and metadata of the first data;

P) a receiver configured to receive a command specifying execution of an unidentified executable on the first data; and

Q) a processor, configured to determine from the metadata of the first data, a content type of the first data, to identify an executable using the content type determined from the metadata, and to operate on the first data using the identified executable.

Rao

Appellant agrees with the Examiner that Rao does not disclose P because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

Appellant agrees with the Examiner that Rao does not disclose Q because, for example, there is no mention of determining a content type from metadata whatsoever. Furthermore, the system used in Rao is for a firmware update and is therefore specific to taking a target firmware update management object and updating it (column 11, lines 21 to 22; column 12, lines 13 to 15). Therefore, the update process is essentially the transfer of known data and Rao does not need to know the content type of the data. Additionally, in Rao, there is no disclosure of what happens to the firmware update data once it has been downloaded and applied (i.e. Rao merely facilitates the transfer of firmware update data and does not discuss the application of firmware). Therefore, Rao uses a system to execute an update process and does not discuss the identification of particular executables for operation on particular content types. Rao does not discuss using metadata from a firmware update, whether for determining an executable or for any other reason. Instead, Rao merely specifies explicit commands to fetch and apply firmware updates.

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of DTD and Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of DTD and/or Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

DTD

More particularly, DTD does not disclose (or suggest) P or Q because there is no method disclosed that uses any commands or execution whatsoever. Furthermore, there is no disclosure of any executables whatsoever.

DTD does not disclose (or suggest) O because there is no discussion of an electronic device comprising a memory.

DTD contains information about SyncML but does not discuss any implementation or uses of SyncML technology. Thus, DTD does not explain how content information in a data element can be used to produce any technical effect.

DTD therefore does not contain any teaching, suggestion or motivation to adapt the information it contains to arrive at the claimed invention. It is respectfully asserted that the TSM test (teaching, suggestion and motivation) provides helpful insights into the non-obviousness of the claims. Moreover, pursuant to *KSR*, there is no reason to modify the

teachings of DTD in an attempt to arrive at the subject claims.

Szeto

Appellant disagrees with the Examiner that Szeto discloses P.

Szeto does not disclose or suggest 'P' because, for example:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the claimed invention, the execution is commanded by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.

   - column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

   Contrastingly, embodiments of the claimed invention use commands sent by a service provider (i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the claimed invention, the executable is determined at the receiving client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

   - column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
   - abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application

identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the invention, the receiving client determines a suitable executable to be used by reading the properties of the metadata of the first data at the receiving client (page 10, line 29 to page 11, line 1).

Szeto does not disclose or suggest 'Q' because:

a) In agreement with the Examiner, there is no disclosure whatsoever of metadata.

b) **In Szeto, the "content type" is specified at the sending client, whereas in the present invention, the content type is determined at the receiving client**. As illustrated in the above diagram at points 3 and 7, the application type (content type) of the data (to be retrieved) in Szeto is specified by the sending client, using an identifier, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client"
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Clearly, in Szeto, the identifier within the IM message provides information about the content type of the IM application and also the location from where the IM application should be retrieved, i.e. the identifier may be a movie trailer identifier that: (i) identifies the IM application to be retrieved as a movie trailer (content type); and (ii) specifies the movie trailer to be retrieved from a movie server (location for retrieval). Therefore the "content type" of the IM application is specified at the sending client. Contrastingly, embodiments of the invention use, for example, a DM client of a mobile phone (receiving client) to determine the content type of the identified first

data from metadata of the first data (page 10, line 29 to page 11, line 1).

c) **In Szeto, the content type is identified (by the sending client) using an identifier, whereas in embodiments of the invention, the content type is determined (by the receiving client) from "metadata of the first data".** As illustrated in the above diagram at point 6, the "first data" (i.e. the data to be executed) in Szeto, requires retrieval from a server and is not present in the IM message. Therefore, as illustrated at point 7 in Szeto, because the content type is determined from the IM message (which does not contain the data to be executed), it cannot be determined from metadata of the first data, i.e.

- column 3, lines 11 to 19: "using the control message [at the receiving client] to retrieve the application from a server"
- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application".

Contrastingly, embodiments of the invention use, for example a DM client of a mobile phone (receiving client) to access the identified data in order to determine the content type of the identified data (page 10, line 29 to page 11, line 1).

d) **In Szeto, the receiving client is told the content type, and hence the executable required, by the sending client**. As illustrated in the above diagram at points 7 and 8, the IM environment at the receiving client uses the identifier in the IM message to select a supporting application, i.e.

- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [in the IM message sent by the sending client], to execute the instant messaging application";
- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";

- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Contrastingly, embodiments of the invention determine a suitable executable to be used by reading the properties of the metadata of the first data at the receiving client (page 10, line 29 to page 11, line 1).

Szeto does not disclose or suggest 'O' because there is no disclosure of metadata of the first data.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to specify execution in one or more mobile phone's using a non-specific command

message. Therefore the control aspect of embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from Appellant's embodiments of the invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

None of Rao, DTD and Szeto individually disclose or suggest the features P and Q of the claimed invention. Therefore Rao, DTD and Szeto, when combined cannot disclose or suggest the features P and Q of the claimed invention.

Alleged Combination of Rao + DTD

The Examiner, in his Advisory Action, comments that Rao is related to SyncML technology and therefore there is motivation to combine it with the SyncML operational specification.

Although the Appellant agrees with the Examiner that DTD may be in a similar field of technology to Rao with regard to the use of SyncML, the Appellant disagrees that these documents can be combined to arrive at the claimed invention or indeed to disclose any additional feature of the claims that DTD or Rao, when taken individually, does not disclose.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. DTD is an information document that discloses a set of mark-up used by SyncML to identify meta-information associated with a SyncML command or data item or collection. Rao specifies named executables to be used when applying a firmware update. DTD provides information only and no implementation/use information. Therefore, the content information determined from metadata (as disclosed in DTD) is not required for the firmware update in Rao. It is incomprehensible why one skilled in the art would use content

type determined from metadata when applying a firmware update, particularly when Rao uses named executables.

Therefore, the combination of these documents does not disclose or suggest the claimed invention. The combination does not disclose or suggest any feature of the claims that the documents do not disclose or suggest individually. Furthermore, there is no motivation to combine the documents because Rao, for example, teaches the use of named executables and there is no requirement to know content type of the firmware update.

Alleged Combination of Rao + Szeto

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, it is incomprehensible why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Furthermore, the combination of these documents does not disclose or suggest any feature of independent claim 19.

Alleged Combination of Szeto + DTD

DTD is only relevant for SyncML technology, and therefore there is no reason to combine its teachings with Szeto. It is unclear why one skilled in the art would use SyncML technology in IM messaging.

The Appellant therefore disagrees with the permissibility of combining Szeto and DTD. It is not known or specified from either document to use an instant messaging application, or indeed an instant message with SyncML technology. Furthermore, as Szeto specifies the content type of the IM application at the sending client, it is unclear why "determining content type from metadata of first data" would be of use.

There is therefore no teaching, suggestion or motivation in Szeto to combine it with DTD whatsoever. Nor is there any reason to do so.

Alleged Combination of Rao + DTD + Szeto

The combination Rao and DTD with Szeto is not obvious because Szeto is not relevant to SyncML technology and Rao and DTD are specific to SyncML technology.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is concerned with firmware updating, and DTD is a Document Type Definition specification for SyncML. The object to be attained in Rao is how to update firmware using SyncML. The object to be attained in DTD is how to use SyncML element types. Szeto does not relate to firmware updates, nor does it relate to using element types in SyncML.

Thus, the combination of Rao and/or DTD with Szeto is considered impermissible, and is therefore non-obvious. One skilled in the art would not look to combine the teachings of these references. Nor would the combination disclose or suggest the claimed invention.

The Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding

application using metadata". Szeto does not identify "the corresponding application" (supporting application) using metadata (as disclosed in DTD) but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore the use of Szeto in combination with Rao and/or DTD would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified data, whereas in embodiments of the claimed invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the claimed invention determines how to do it. Thus, Szeto teaches away from the claimed invention by specifying the control of received messages at the receiving client and not at the sending client.

Accordingly, for at least the foregoing reasons, independent claim 19 is considered to be new and non-obvious in view of the cited art. Similarly, claims 20-21, although containing their own allowable subject matter, also are considered to be new and non-obvious at least in view of their dependency from an allowable independent claim.

Claim 34

Independent claim 34 currently recites:

R) An electronic device, comprising:
means for storing first data;

S) means for receiving a command specifying execution of an unidentified executable on the first data;

T) means for determining, from metadata, a content type of the identified first data;

U) means for identifying an executable using the content type determined from the metadata; and

V) means for operating on the identified data using the identified executable.

The "means plus function" may, for example, be interpreted as follows:

Means plus function	Example in specification (example, figure reference numeral)	Specification reference (page; line)
"means for storing"	Memory, 13	5; 27
"means for receiving"	Cellular radio transceiver, 12	5; 27
"means for determining"	Processor, 11	5; 27
"means for identifying"	Processor, 11	5; 27
"means for operating"	Processor, 11	5; 27

Rao

Appellant agrees with the Examiner that Rao does not disclose S because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

Appellant agrees with the Examiner that Rao does not disclose T because, for example, there is no mention of determining a content type from metadata whatsoever. Furthermore, the system used in Rao is for a firmware update and is therefore specific to taking a target firmware update management object and updating it (column 11, lines 21 to 22; column 12, lines 13 to 15). Therefore the update process is essentially the transfer of known data and Rao does not need to know the content type of the data.

Appellant agrees with the Examiner that Rao does not disclose U because, for example, there is no mention of determining a content type from metadata whatsoever, as explained above. Furthermore, in Rao, the execution is of an update process and there is no disclosure of what happens to the firmware update data once it has been downloaded and applied (i.e. Rao

merely facilitates the transfer of firmware update data and does not discuss the application of firmware). Therefore Rao uses a system to execute an update process and does not discuss the identification of particular executables for operation on particular content types. Rao does not discuss using metadata from a firmware update, whether for determining an executable or for any other reason. Instead, Rao merely specifies explicit commands to fetch and apply firmware updates.

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of DTD and Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of DTD and/or Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

DTD

DTD does not disclose (or suggest) S because there is no method disclosed that uses any commands or execution whatsoever.

DTD does not disclose (or suggest) U because there is no disclosure of any executables whatsoever.

DTD does not disclose (or suggest) V because there is no execution or executable disclosed whatsoever.

DTD does not disclose (or suggest) R because there is no discussion of an electronic device comprising a means for storing first data.

DTD contains information about SyncML but does not discuss any implementation or uses of SyncML technology. Thus, DTD does not explain how content information in a data element can be used to produce any technical effect.

DTD therefore does not contain any teaching, suggestion or motivation to adapt the information it contains to arrive at the claimed invention. It is respectfully asserted that the TSM test (teaching, suggestion and motivation) provides helpful insights into the non-obviousness of the claims. Moreover, pursuant to *KSR*, there is no reason to modify the teachings of DTD in an attempt to arrive at the subject claims.

Szeto

Appellant disagrees with the Examiner that Szeto discloses S.

Szeto does not disclose or suggest 'S' because:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the claimed invention, the execution is commanded by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.

   - column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

   Contrastingly, embodiments of the invention use commands sent by a service provider (i.e. sending client), to specify execution of identified data within a mobile phone (i.e.

receiving client) (page 10, lines 11 to 12).

b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the invention, the executable is determined at the receiving client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the invention, the receiving client determines a suitable executable to be used by reading the properties of the metadata of the first data at the receiving client (page 10, line 29 to page 11, line 1).

Szeto does not disclose or suggest 'T' because:

a) In agreement with the Examiner, there is no disclosure whatsoever of metadata.

b) **In Szeto, the "content type" is specified at the sending client, whereas in the present invention, the content type is determined at the receiving client.** As illustrated in the above diagram at points 3 and 7, the application type (content type) of the data (to be retrieved) in Szeto is specified by the sending client, using an identifier, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client"

- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Clearly, in Szeto, the identifier within the IM message provides information about the content type of the IM application and also the location from where the IM application should be retrieved, i.e. the identifier may be a movie trailer identifier that: (i) identifies the IM application to be retrieved as a movie trailer (content type); and (ii) specifies the movie trailer to be retrieved from a movie server (location for retrieval). Therefore the "content type" of the IM application is specified at the sending client. Contrastingly, embodiments of the invention use, for example, a DM client of a mobile phone (receiving client) to determine the content type of the identified first data from metadata of the first data (page 10, line 29 to page 11, line 1).

c) **In Szeto, the content type is identified (by the sending client) using an identifier, whereas in embodiments of the invention, the content type is determined (by the receiving client) from "first data".** As illustrated in the above diagram at point 6, the "first data" (i.e. the data to be executed) in Szeto, requires retrieval from a server and is not present in the IM message. Therefore, as illustrated at point 7 in Szeto, because the content type is determined from the IM message (which does not contain the data to be executed), it cannot be determined from first data, i.e.

- column 3, lines 11 to 19: "using the control message [at the receiving client] to retrieve the application from a server"
- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application".

Contrastingly, embodiments of the invention use, for example a DM client of a mobile phone (receiving client) to access the identified data in order to determine the content type of the identified data (page 10, line 29 to page 11, line 1).

Szeto does not disclose or suggest 'U' because:

a) As above, there is no disclosure of metadata.

b) As above, the content type is specified by the sending client and not determined at the receiving client.

c) As above, the content type is specified using an identifier and is not determined from first data.

d) **In Szeto, the receiving client is told the content type, and hence the executable required, by the sending client**. As illustrated in the above diagram at points 7 and 8, the IM environment at the receiving client uses the identifier in the IM message to select a supporting application.

   - Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [in the IM message sent by the sending client], to execute the instant messaging application";
   - column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
   - column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
   - column 13, line 22: "A movie trailer identifier is passed to the movie server..."

   Contrastingly, embodiments of the invention determine a suitable executable to be used by reading the properties of the metadata of the first data at the receiving client (page 10, line 29 to page 11, line 1).

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment'

that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to specify execution in one or more mobile phone's using a non-specific command message. Therefore the control aspect of embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

None of Rao, DTD and Szeto individually disclose or suggest the features S and U of the claimed invention. Therefore Rao, DTD and Szeto, when combined cannot disclose or suggest the features S and U of the claimed invention.

Alleged Combination of Rao + DTD

The Examiner, in his Advisory Action, comments that Rao is related to SyncML technology and therefore there is motivation to combine it with the SyncML operational specification.

Although the Appellant agrees with the Examiner that DTD may be in a similar field of technology to Rao with regard to the use of SyncML, the Appellant disagrees that these documents can be combined to arrive at the claimed invention or indeed to disclose any additional feature of the claims that DTD or Rao, when taken individually, does not disclose

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. DTD is an information document that discloses a set of mark-up used by SyncML to identify meta-information associated with a SyncML command or data item or collection. Rao specifies named executables to be used when applying a firmware update. DTD provides information only and no implementation/use information. Therefore, the content information determined from metadata (as disclosed in DTD) is not required for the firmware update in Rao. It is incomprehensible why one skilled in the art would use content type determined from metadata when applying a firmware update, particularly when Rao uses named executables.

Therefore, the combination of these documents does not disclose or suggest the claimed invention. The combination does not disclose or suggest any feature of the claims that the documents do not disclose or suggest individually. Furthermore, there is no motivation to combine the documents because Rao teaches the use of named executables and there is no requirement to know content type of the firmware update.

Alleged Combination of Rao + Szeto

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving

IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, it is incomprehensible why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Nor is there any reason to do so. Furthermore, the combination of these documents would not result in the features of independent claim 34.

Alleged Combination of Szeto + DTD

DTD is only relevant for SyncML technology, and therefore it is incomprehensible why the Examiner has combined this document with Szeto. It is unclear why one skilled in the art would use SyncML technology in IM messaging.

The Appellant therefore disagrees with the permissibility of combining Szeto and DTD. It is not known or specified from either document to use an instant messaging application, or indeed an instant message with SyncML technology. Furthermore, as Szeto specifies the content type of the IM application at the sending client, it is unclear why "determining content type from metadata of first data" would be of use.

There is therefore no teaching, suggestion or motivation in Szeto to combine it with DTD whatsoever. Nor is there any reason to do so.

Alleged Combination of Rao + DTD + Szeto

The combination Rao and DTD with Szeto is not obvious because Szeto is not relevant to SyncML technology and Rao and DTD are specific to SyncML technology. Thus, there is no reason to combine and modify these documents.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is concerned with firmware updating, and DTD is a Document Type Definition specification for SyncML. The object to be attained in Rao is how to update firmware using SyncML. The object to be attained in DTD is how to use SyncML element types. Szeto does not relate to firmware updates, nor does it relate to using element types in SyncML.

Thus, the combination of Rao and/or DTD with Szeto is considered impermissible, and is therefore non-obvious. One skilled in the art would not look to combine the teachings of these references. Nor would the combination disclose or suggest the claimed invention.

The Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata". Szeto does not identify "the corresponding application" (supporting application) using metadata (as disclosed in DTD) but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore the use of Szeto in combination with Rao and/or DTD would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified

data, whereas in the present invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the claimed invention determines how to do it. Thus, Szeto teaches away from the claimed invention by specifying the control of received messages at the receiving client and not at the sending client.

Accordingly, for at least the foregoing reasons, independent claim 34 is believed to be new and non-obvious in view of the cited art.

Claim 41

Independent claim 41 currently recites:

W) A computer program product comprising program instructions embodied on a tangible computer-readable medium, execution of the program instructions resulting in operations comprising:

X) automatically determining, from metadata of first data, a content type of the first data;

Y) automatically identifying an executable using the content type determined from the metadata; and

Z) enabling the first data to be operated on using the identified executable.

Rao

Appellant agrees with the Examiner that Rao does not disclose X because, for example, there is no mention of determining a content type from metadata whatsoever. Furthermore, the system used in Rao is for a firmware update and is therefore specific to taking a target firmware update management object and updating it (column 11, lines 21 to 22; column 12, lines 13 to 15). Therefore, the update process is essentially the transfer of known data and Rao does not need to know the content type of the data.

Appellant agrees with the Examiner that Rao does not disclose Y because, for example, there is no mention of determining a content type from metadata whatsoever, as explained above. Furthermore, in Rao, the execution is of an update process and there is no disclosure of what

happens to the firmware update data once it has been downloaded and applied (i.e. Rao merely facilitates the transfer of firmware update data and does not discuss the application of firmware). Therefore, Rao uses a system to execute an update process and does not discuss the identification of particular executables for operation on particular content types. Rao does not discuss using metadata from a firmware update, whether for determining an executable or for any other reason. Instead, Rao merely specifies explicit commands to fetch and apply firmware updates.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device and specifies named executables to be used when applying a firmware update. Therefore, content information determined from metadata is not required for the firmware update in Rao. There is no reason why one skilled in the art would use content type determined from metadata when applying a firmware update, particularly when Rao uses named executables.

DTD

DTD does not disclose (or suggest) Y because there is no disclosure of any executables whatsoever.

DTD does not disclose (or suggest) Z because there is no execution or executable disclosed whatsoever.

DTD does not disclose (or suggest) W because there is no disclosure whatsoever of any computer program product comprising computer program instructions embodied on a tangible computer-readable medium.

DTD contains information about SyncML but does not discuss any implementation or uses of SyncML technology. Thus, DTD does not explain how content information in a data element can be used to produce any technical effect.

DTD therefore does not contain any teaching, suggestion or motivation to adapt the information it contains to arrive at the claimed invention. It is respectfully asserted that the TSM test (teaching, suggestion and motivation) provides helpful insights into the non-obviousness of the claims. Moreover, pursuant to *KSR*, there is no reason to modify the teachings of DTD in an attempt to arrive at the subject claims.

Szeto

Szeto does not disclose or suggest 'X' because, for example:

a) In agreement with the Examiner, there is no disclosure whatsoever of metadata.

b) **In Szeto, the "content type" is specified at the sending client, whereas in the present invention, the content type is determined at the receiving client**. As illustrated in the above diagram at points 3 and 7, the application type (content type) of the data (to be retrieved) in Szeto is specified by the sending client, using an identifier, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client"
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Clearly, in Szeto, the identifier within the IM message provides information about the content type of the IM application and also the location from where the IM application should be retrieved, i.e. the identifier may be a movie trailer identifier that: (i) identifies the IM application to be retrieved as a movie trailer (content type); and (ii) specifies the movie trailer to be retrieved from a movie server (location for retrieval). Therefore the "content type" of the IM application is specified at the sending client. Contrastingly, embodiments of the invention use, for example, a DM client (i.e.

computer program) of a mobile phone (receiving client) to determine the content type of the identified first data from metadata of the first data (page 10, line 29 to page 11, line 1).

c) **In Szeto, the content type is identified (by the sending client) using an identifier, whereas in the present invention, the content type is determined (by the receiving client) from "metadata of first data".** As illustrated in the above diagram at point 6, the "first data" (i.e. the data to be executed) in Szeto, requires retrieval from a server and is not present in the IM message. Therefore, as illustrated at point 7 in Szeto, because the content type is determined from the IM message (which does not contain the data to be executed), it cannot be determined from metadata of first data.

- column 3, lines 11 to 19: "using the control message [at the receiving client] to retrieve the application from a server"
- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application".

Contrastingly, embodiments of the invention use, for example a DM client (i.e. computer program) of a mobile phone (receiving client) to access the identified data in order to determine the content type of the identified data (page 10, line 29 to page 11, line 1).

Szeto does not disclose or suggest 'Y' because, for example:

e) As above, there is no disclosure of metadata.

f) As above, the content type is specified by the sending client and not determined at the receiving client.

g) As above, the content type is specified using an identifier and is not determined from first data.

h) **In Szeto, the receiving client is told the content type, and hence the executable required, by the sending client**. As illustrated in the above diagram at points 7 and

8, the IM environment at the receiving client uses the identifier in the IM message to select a supporting application.

- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [in the IM message sent by the sending client], to execute the instant messaging application";
- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Contrastingly, embodiments of the invention determine a suitable executable to be used by reading the properties of the metadata of the first data at the receiving client (page 10, line 29 to page 11, line 1).

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore told the content type of the data to be retrieved and hence are given information about the executable required.

Embodiments of the invention are directed to the receiving client's (e.g. service provider) ability to identify an appropriate executable for use on first data by analyzing metadata of the first data. Therefore, the receiving client of the invention may receive a non-specific command to execute data (i.e. receiving client is not told any information about the executable required) and may determine an appropriate executable to be used.

Thus, Szeto teaches away from the claimed invention by specifying information about the executable to be used on data being retrieved. Contrastingly, embodiments of the invention allow the receiving client to determine what executable should be used.

Combination Problems

None of Rao, DTD and Szeto individually disclose or suggest the feature Y of the claimed invention. Therefore, Rao, DTD and Szeto, when combined cannot disclose or suggest the feature Y of the claimed invention. Nor is there any reason to combine and modify the teachings of the references in an attempt to arrive at the subject claims.

Alleged Combination of Rao + DTD

The Examiner, in his Advisory Action, comments that Rao is related to SyncML technology and therefore there is motivation to combine it with the SyncML operational specification.

Although the Appellant agrees with the Examiner that DTD may be in a similar field of technology to Rao with regard to the use of SyncML, the Appellant disagrees that these documents can be combined in an attempt to arrive at the claimed invention or indeed to disclose or suggest any additional feature of the claims that DTD or Rao, when taken individually, does not disclose or suggest.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. DTD is an information document that discloses a set of mark-up used by SyncML to identify meta-information associated with a SyncML command or data item or collection. Rao specifies named executables to be used when applying a firmware update. DTD provides information only and no implementation/use information. Therefore, the

content information determined from metadata (as disclosed in DTD) is not required for the firmware update in Rao. There is no reason why one skilled in the art would use content type determined from metadata when applying a firmware update, particularly when Rao uses named executables.

Therefore, the combination of these documents does not disclose or suggest the claimed invention. The combination does not disclose or suggest any feature of the claims that the documents do not disclose or suggest individually. Furthermore, there is no motivation to combine the documents because Rao teaches the use of named executables and there is no requirement to know content type of the firmware update.

Alleged Combination of Rao + Szeto

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, there is no reason why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Furthermore, the combination of these documents does not disclose or suggest the features of

independent claim 41.

Alleged Combination of Szeto + DTD

DTD is only relevant for SyncML technology, and therefore there is no reason to combine its teachings with Szeto. It is unclear why one skilled in the art would use SyncML technology in IM messaging.

The Appellant therefore disagrees with the permissibility of combining Szeto and DTD. It is not known or specified from either document to use an instant messaging application, or indeed an instant message with SyncML technology. Furthermore, as Szeto specifies the content type of the IM application at the sending client, it is unclear why "determining content type from metadata of first data" would be of use.

There is therefore no teaching, suggestion or motivation in Szeto to combine it with DTD whatsoever. Nor is there any reason to do so.

Alleged Combination of Rao + DTD + Szeto

The combination Rao and DTD with Szeto is not obvious because Szeto is not relevant to SyncML technology and Rao and DTD are specific to SyncML technology.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is concerned with firmware updating, and DTD is a Document Type Definition specification for SyncML. The object to be attained in Rao is how to update firmware using SyncML. The object to be attained in DTD is how to use SyncML element types. Szeto does not relate to firmware updates, nor does it relate to using element types in SyncML.

Thus, the combination of Rao and/or DTD with Szeto is considered impermissible, and is therefore non-obvious. One skilled in the art would not look to combine the teachings of these references. Nor would the combination disclose or suggest the claimed invention.

The Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata". Szeto does not identify "the corresponding application" (supporting application) using metadata (as disclosed in DTD) but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore the use of Szeto in combination with Rao and/or DTD would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified data, whereas in the present invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the invention merely determines how to do it. Thus, Szeto teaches away from the claimed invention by specifying the control of received messages at the receiving client and not at the sending client.

Accordingly, for at least the foregoing reasons, independent claim 41 is believed to be new and non-obvious in view of the cited art.

Claim 43

Independent claim 43 currently recites:

AA) A method, comprising:
receiving a first command at an electronic device, the first command specifying creation of a leaf node in a hierarchical data structure, and identifying first data to be stored at the leaf node and metadata indicating a content type of the first data;

BB) creating the leaf node at the electronic device;

CC) receiving a second command, at the electronic device, that specifies execution of an unidentified executable on the first data stored at the created leaf node;

DD) determining, from the metadata, the content type of the first data;

EE) identifying an executable using the content type determined from the metadata; and

FF) operating on the first data using the identified executable.

Rao

Appellant agrees with the Examiner that Rao does not disclose CC because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

Appellant agrees with the Examiner that Rao does not disclose DD because, for example, there is no mention of determining a content type from metadata whatsoever. Furthermore, the system used in Rao is for a firmware update and is therefore specific to taking a target firmware update management object and updating it (column 11, lines 21 to 22; column 12, lines 13 to 15). Therefore, the update process is essentially the transfer of known data and Rao does not need to know the content type of the data.

The Appellant agrees with the Examiner that Rao does not disclose EE because, for example, there is no mention of determining a content type from metadata whatsoever, as explained above. Furthermore, in Rao, the execution is of an update process and there is no disclosure of what happens to the firmware update data once it has been downloaded and applied (i.e. Rao merely facilitates the transfer of firmware update data and does not discuss the application of firmware). Therefore, Rao uses a system to execute an update process and does not discuss the identification of particular executables for operation on particular content types. Rao does not discuss using metadata from a firmware update, whether for determining an executable or for any other reason. Instead, Rao merely specifies explicit commands to

fetch and apply firmware updates.

Appellant agrees with the Examiner that Rao does not disclose AA because, for example, there is no mention of metadata indicating content type.

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of DTD and Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of DTD and/or Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

DTD

DTD does not disclose (or suggest) CC because there is no method disclosed that uses any commands or execution whatsoever.

DTD does not disclose (or suggest) EE because there is no disclosure of any executables whatsoever.

DTD does not disclose (or suggest) FF because there is no execution or executable disclosed whatsoever.

DTD does not disclose (or suggest) AA because there is no method disclosed that uses any

commands or execution whatsoever. Also, there is no disclosure of any commands nor is there any disclosure of creation of a hierarchical nodular data structure.

DTD does not disclose (or suggest) BB because there is no disclosure of creation of a leaf node at an electronic device.

DTD contains information about SyncML, but does not discuss any implementation or uses of SyncML technology. Thus, DTD does not explain how content information in a data element can be used to produce any technical effect.

DTD therefore does not contain any teaching, suggestion or motivation to adapt the information it contains to arrive at the claimed invention. It is respectfully asserted that the TSM test (teaching, suggestion and motivation) provides helpful insights into the non-obviousness of the claims. Moreover, pursuant to *KSR*, there is no reason to modify the teachings of DTD in an attempt to arrive at the subject claims.

Szeto

Appellant disagrees with the Examiner that Szeto discloses CC.

Szeto does not disclose 'CC' because, for example:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the claimed invention, the execution is commanded by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.

- column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

Contrastingly, embodiments of the invention use commands sent by a service provider

(i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the claimed invention, the executable is determined at the receiving client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the invention, the receiving client determines a suitable executable to be used by reading the properties of the metadata at the receiving client (page 10, line 29 to page 11, line 1).

c) There is no disclosure whatsoever in Szeto of creation of a leaf node or a hierarchical nodular data structure or first data stored at the created first leaf node.

Szeto does not disclose 'DD' because, for example:

a) In agreement with the Examiner, there is no disclosure whatsoever of metadata.

b) **In Szeto, the "content type" is specified at the sending client, whereas in embodiments of the invention, the content type is determined at the receiving client**. As illustrated in the above diagram at points 3 and 7, the application type (content type) of the data (to be retrieved) in Szeto is specified by the sending client, using an identifier, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client"

- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Clearly, in Szeto, the identifier within the IM message provides information about the content type of the IM application and also the location from where the IM application should be retrieved, i.e. the identifier may be a movie trailer identifier that: (i) identifies the IM application to be retrieved as a movie trailer (content type); and (ii) specifies the movie trailer to be retrieved from a movie server (location for retrieval). Therefore the "content type" of the IM application is specified at the sending client. Contrastingly, embodiments of the present invention use, for example, a DM client of a mobile phone (receiving client) to determine the content type of the identified first data from metadata (page 10, line 29 to page 11, line 1).

c) **In Szeto, the content type is identified (by the sending client) using an identifier, whereas in the present invention, the content type is determined (by the receiving client) from metadata of the first data.** As illustrated in the above diagram at point 6, the "first data" (i.e. the data to be executed) in Szeto, requires retrieval from a server and is not present in the IM message. Therefore, as illustrated at point 7 in Szeto, because the content type is determined from the IM message (which does not contain the data to be executed), it cannot be determined from metadata stored at the first leaf node, i.e.

- column 3, lines 11 to 19: "using the control message [at the receiving client] to retrieve the application from a server"
- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application".

Contrastingly, embodiments of the invention use, for example a DM client of a mobile phone (receiving client) to access the identified data in order to determine the content

type of the identified data (page 10, line 29 to page 11, line 1).

d) There is no disclosure whatsoever in Szeto of a leaf node.

Szeto does not disclose or suggest 'EE' because, for example:

a) As above, there is no disclosure of metadata.

b) As above, the content type is specified by the sending client and not determined at the receiving client.

c) As above, the content type is specified using an identifier and is not determined from metadata.

d) **In Szeto, the receiving client is told the content type, and hence the executable required, by the sending client**. As illustrated in the above diagram at points 7 and 8, the IM environment at the receiving client uses the identifier in the IM message to select a supporting application, i.e.

- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [in the IM message sent by the sending client], to execute the instant messaging application";
- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Contrastingly, embodiments of the invention determine a suitable executable to be used by reading the properties of the metadata stored at the created leaf node at the receiving client (page 10, line 29 to page 11, line 1).

e) There is no disclosure whatsoever in Szeto of a leaf node.

Szeto does not disclose or suggest AA because there is no disclosure whatsoever of a command specifying creation of a leaf node nor is there any disclosure of first data stored at the identified first leaf node. There is no disclosure in Szeto of metadata.

Szeto does not disclose or suggest BB because there is no disclosure whatsoever of a creating a leaf node at the receiving client.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to specify execution in one or more mobile phone's using a non-specific command message. Therefore the control aspect of the present invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

None of Rao, DTD and Szeto individually disclose or suggest the features AA, CC and EE of the claimed invention. Therefore, Rao, DTD and Szeto, when combined cannot disclose or suggest the features AA, CC and EE of the claimed invention. Nor is there any reason to combine and modify the teachings of these references in an attempt to arrive at the subject claims.

Alleged Combination of Rao + DTD

The Examiner, in his Advisory Action, comments that Rao is related to SyncML technology and therefore there is motivation to combine it with the SyncML operational specification.

Although the Appellant agrees with the Examiner that DTD may be n a similar field of technology to Rao with regard to the use of SyncML, the Appellant disagrees that these documents can be combined in an attempt to arrive at the claimed invention or indeed to disclose or suggest any additional feature of the claims that DTD or Rao, when taken individually, does not disclose or suggest.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. DTD is an information document that discloses a set of mark-up used by SyncML to identify meta-information associated with a SyncML command or data item or collection. Rao specifies named executables to be used when applying a firmware update. DTD provides information only and no implementation/use information. Therefore, the content information determined from metadata (as disclosed in DTD) is not required for the firmware update in Rao. It is incomprehensible why one skilled in the art would use content type determined from metadata when applying a firmware update, particularly when Rao uses

named executables.

Therefore, the combination of these documents does not disclose or suggest the claimed invention. The combination does not disclose or suggest any feature of the claims that the documents do not disclose or suggest individually. Furthermore, there is no motivation to combine the documents because Rao teaches the use of named executables and there is no requirement to know content type of the firmware update.

Alleged Combination of Rao + Szeto

Szeto relates to controlling an application in a shared environment which exists between two or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. There is no reason why one skilled in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Furthermore, the combination of these documents does not disclose or suggest the features of independent claim 43.

Alleged Combination of Szeto + DTD

DTD is only relevant for SyncML technology, and there is no reason to combine this document with Szeto. It is unclear why one skilled in the art would use SyncML technology in IM messaging.

The Appellant therefore disagrees with the permissibility of combining Szeto and DTD. It is not known or specified from either document to use an instant messaging application, or indeed an instant message with SyncML technology. Furthermore, as Szeto specifies the content type of the IM application at the sending client, it is unclear why "determining, from metadata, the content type of the first data" would be of use.

There is therefore no teaching, suggestion or motivation in Szeto to combine it with DTD whatsoever. Nor is there any reason to do so.

Alleged Combination of Rao + DTD + Szeto

There would thus be no reason to combine Rao and DTD with Szeto because, for example, Szeto is not relevant to SyncML technology and Rao and DTD are specific to SyncML technology.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is concerned with firmware updating, and DTD is a Document Type Definition specification for SyncML. The object to be attained in Rao is how to update firmware using SyncML. The object to be attained in DTD is how to use SyncML element types. Szeto does not relate to firmware updates, nor does it relate to using element types in SyncML.

Thus, the combination of Rao or DTD with Szeto is considered impermissible, and is therefore non-obvious. One skilled in the art would not look to combine the teachings of these references. Nor would the combination disclose or suggest the claimed invention.

The Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata". Szeto does not identify "the corresponding application"

(supporting application) using metadata (as disclosed in DTD) but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore the use of Szeto in combination with Rao and/or DTD would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified data, whereas in the present invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the invention merely determines how to do it. Thus, Szeto teaches away from the claimed invention by specifying the control of received messages at the receiving client and not at the sending client.

Accordingly, for at least the foregoing reasons, independent claim 43 is believed to be new and non-obvious in view of the cited art.

Claim 44

Independent claim 44 currently recites:

GG) An electronic device, comprising:
a receiver configured to receive a first command at an electronic device, the first command specifying creation of a leaf node in a hierarchical data structure, and identifying first data to be stored at the leaf node and metadata indicating a content type of the first data; and

HH) a processor configured to create the leaf node at the electronic device, wherein

II) the receiver is further configured to receive a second command that specifies execution of an unidentified executable on the first data stored at the created leaf node, and the processor is further configured to determine, from the metadata, a content type of the first data, to identify an executable using the content type determined from the metadata, and to operate on the first data using the identified executable.

Rao

Appellant agrees with the Examiner that Rao does not disclose II because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update). Furthermore, there is no mention of determining a content type from metadata whatsoever. The system used in Rao is for a firmware update and is therefore specific to taking a target firmware update management object and updating it (column 11, lines 21 to 22; column 12, lines 13 to 15). Therefore the update process is essentially the transfer of known data and Rao does not need to know the content type of the data.

Furthermore, in Rao, the execution is of an update process and there is no disclosure of what happens to the firmware update data once it has been downloaded and applied (i.e. Rao merely facilitates the transfer of firmware update data and does not discuss the application of firmware). Therefore Rao uses a system to execute an update process and does not discuss the identification of particular executables for operation on particular content types. Rao does not discuss using metadata from a firmware update, whether for determining an executable or for any other reason. Instead, Rao merely specifies explicit commands to fetch and apply

firmware updates.

Appellant also agrees with Examiner that Rao does not disclose GG because, for example, there is no mention of metadata indicating content type.

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of DTD and Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of DTD and/or Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

DTD

DTD does not disclose (or suggest) II because there is no method disclosed that uses any commands, executables or execution whatsoever.

DTD does not disclose (or suggest) GG because there is no method disclosed that uses any commands or execution whatsoever. Also, there is no disclosure of any commands nor is there any disclosure of creation of a hierarchical nodular data structure.

DTD does not disclose (or suggest) HH because there is no disclosure of creation of a leaf node at an electronic device.

DTD contains information about SyncML, but does not discuss any implementation or uses of SyncML technology. Thus, DTD does not explain how content information in a data element can be used to produce any technical effect.

DTD therefore does not contain any teaching, suggestion or motivation to adapt the information it contains to arrive at the claimed invention. It is respectfully asserted that the TSM test (teaching, suggestion and motivation) provides helpful insights into the non-obviousness of the claims. Moreover, pursuant to *KSR*, there is no reason to modify the teachings of DTD in an attempt to arrive at the subject claims.

Szeto

Appellant respectfully disagrees with the Examiner that Szeto discloses II.

Szeto does not disclose or suggest 'II' because, for example:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the invention, the execution is commanded by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.
   - column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

   Contrastingly, embodiments of the invention use commands sent by a service provider (i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).
b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the invention, the executable is determined at the receiving client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the invention, the receiving client determines a suitable executable to be used by reading the properties of the metadata at the receiving client (page 10, line 29 to page 11, line 1).

c) In agreement with the Examiner, there is no disclosure whatsoever of metadata.

d) **In Szeto, the "content type" is specified at the sending client, whereas in embodiments of the invention, the content type is determined at the receiving client**. As illustrated in the above diagram at points 3 and 7, the application type (content type) of the data (to be retrieved) in Szeto is specified by the sending client, using an identifier, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client"
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Clearly, in Szeto, the identifier within the IM message provides information about the content type of the IM application and also the location from where the IM application should be retrieved, i.e. the identifier may be a movie trailer identifier that: (i) identifies the IM application to be retrieved as a movie trailer (content type); and (ii) specifies the movie trailer to be retrieved from a movie server (location for retrieval). Therefore the "content type" of the IM application is specified at the sending client.

Contrastingly, embodiments of the invention use, for example, a DM client of a mobile phone (receiving client) to determine the content type of the identified first data from metadata (page 10, line 29 to page 11, line 1).

e) **In Szeto, the content type is identified (by the sending client) using an identifier, whereas in the present invention, the content type is determined (by the receiving client) from metadata of the first data.** As illustrated in the above diagram at point 6, the "first data" (i.e. the data to be executed) in Szeto, requires retrieval from a server and is not present in the IM message. Therefore, as illustrated at point 7 in Szeto, because the content type is determined from the IM message (which does not contain the data to be executed), it cannot be determined from metadata stored at the created leaf node, i.e.

- column 3, lines 11 to 19: "using the control message [at the receiving client] to retrieve the application from a server"
- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application".

Contrastingly, embodiments of the invention use, for example a DM client of a mobile phone (receiving client) to access the identified data in order to determine the content type of the identified data (page 10, line 29 to page 11, line 1).

f) **In Szeto, the receiving client is told the content type, and hence the executable required, by the sending client**. As illustrated in the above diagram at points 7 and 8, the IM environment at the receiving client uses the identifier in the IM message to select a supporting application, i.e.

- Abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [in the IM message sent by the sending client], to execute the instant messaging application";

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- column 13, lines 3 to 6: "From the [identifier of the] IM message, the IM client 202 determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step 1204";
- column 13, line 22: "A movie trailer identifier is passed to the movie server..."

Contrastingly, embodiments of the invention determine a suitable executable to be used by reading the properties of the metadata stored at the first leaf node at the receiving client (page 10, line 29 to page 11, line 1).

g) There is no disclosure whatsoever in Szeto of a leaf node.

Szeto does not disclose or suggest GG because, for example, there is no disclosure whatsoever of a command specifying creation of a leaf node nor is there any disclosure of first data stored at the identified first leaf node. There is no disclosure in Szeto of metadata.

Szeto does not disclose or suggest HH because there is no disclosure whatsoever of a creating a leaf node at the receiving client.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and

the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to create a leaf node storing first data and metadata of the first data at one or more mobile phones (receiving client) and specifying execution using a non-specific command message. Therefore the control aspect of embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

None of Rao, DTD and Szeto individually disclose or suggest, for example, the features GG and II of the claimed invention. Therefore, Rao, DTD and Szeto, when combined cannot disclose or suggest the features GG and II of the claimed invention. Nor is there any reason to combine and modify the teachings of these references in an attempt to arrive at the subject claims.

Alleged Combination of Rao + DTD

The Examiner, in his Advisory Action, comments that Rao is related to SyncML technology and therefore there is motivation to combine it with the SyncML operational specification.

Although the Appellant agrees with the Examiner that DTD may be in a similar field of technology to Rao with regard to the use of SyncML, the Appellant disagrees that these

documents can be combined in an attempt to arrive at the present invention or indeed to disclose any additional feature of the claims that DTD or Rao, when taken individually, does not disclose.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. DTD is an information document that discloses a set of mark-up used by SyncML to identify meta-information associated with a SyncML command or data item or collection. Rao specifies named executables to be used when applying a firmware update. DTD provides information only and no implementation/use information. Therefore, the content information determined from metadata (as disclosed in DTD) is not required for the firmware update in Rao. It is incomprehensible why one skilled in the art would use content type determined from metadata when applying a firmware update, particularly when Rao uses named executables.

Therefore, the combination of these documents does not disclose or suggest the claimed invention. The combination does not disclose any feature of the claims that the documents do not disclose individually. Furthermore, there is no motivation to combine the documents because Rao teaches the use of named executables and there is no requirement to know content type of the firmware update.

Alleged Combination of Rao + Szeto

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM

messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, there is no reason why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Furthermore, the combination of these documents does not disclose the features of independent claim 44.

Alleged Combination of Szeto + DTD

DTD is only relevant for SyncML technology, and therefore there is no reason why one skilled in the art would combine its teachings with Szeto. It is unclear why one skilled in the art would use SyncML technology in IM messaging.

The Appellant therefore disagrees with the permissibility of combining Szeto and DTD. It is not known or specified from either document to use an instant messaging application, or indeed an instant message with SyncML technology. Furthermore, as Szeto specifies the content type of the IM application at the sending client, it is unclear why "determine, from the metadata, the content type of the first data" would be of use.

There is therefore no teaching, suggestion or motivation in Szeto to combine it with DTD whatsoever. Nor is there any reason to do so.

Alleged Combination of Rao + DTD + Szeto

There would thus be no reason to combine Rao and DTD with Szeto because, for example, Szeto is not relevant to SyncML technology and Rao and DTD are specific to SyncML technology.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is

concerned with firmware updating, and DTD is a Document Type Definition specification for SyncML. The object to be attained in Rao is how to update firmware using SyncML. The object to be attained in DTD is how to use SyncML element types. Szeto does not relate to firmware updates, nor does it relate to using element types in SyncML.

Thus, the combination of Rao or DTD with Szeto is considered impermissible, and is therefore non-obvious. One skilled in the art would not look to combine the teachings of these references. Nor would the combination disclose or suggest the claimed invention.

The Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata". Szeto does not identify "the corresponding application" (supporting application) using metadata (as disclosed in DTD) but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore the use of Szeto in combination with Rao and/or DTD would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified data, whereas in the present invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the claimed invention merely determines how to do it. Thus, Szeto teaches away from the claimed invention by specifying the control of received messages at the receiving client and not at the sending client.

Rejection under 35 U.S.C. 103(a) based on US Patent 6,978,453 (Rao) in view of US Patent no. 7,188,143 (Szeto)

Claims 22 to 23 and 25 to 28

Independent claim 22 currently recites:

JJ) A data structure embodied on a computer-readable medium, comprising:

KK) code identifying first data and specifying execution of an unidentified executable on the first data.

Rao

Appellant agrees with the Examiner that Rao does not disclose KK because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

Appellant respectfully disagrees with the Examiner that Rao discloses JJ because, for example, there is no disclosure of any data structure embodied on a computer-readable medium, for example, a smart card (page 7, lines 10 to 11).

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

Szeto

Appellant respectfully disagrees with the Examiner that Szeto discloses KK.

Szeto does not disclose or suggest 'KK' because, for example:

a) **In Szeto, the execution is specified by the receiving client, whereas in embodiments of the invention, the execution is specified by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is specified by the IM environment of the receiving client and not the IM message, i.e.

   - column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

   Contrastingly, embodiments of the invention use a data structure embodied on a computer-readable medium (i.e. a smart card as a sending client – page 7, lines 10 to 11), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the invention, the executable is unidentified at the sending client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

   - column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
   - abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

   Contrastingly, in embodiments of the invention, the data structure embodied on a

computer-readable medium does not contain any information about the executable.

Szeto does not disclose or suggest JJ because, for example, there is no disclosure of any data structure embodied on a computer-readable medium.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. smart card – page 7, lines 10 to 11) ability to specify execution in one or more mobile phone's using a non-specific command message. Therefore the control aspect of embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message

at the receiving client should occur.

Combination Problems

Rao nor Szeto individually disclose or suggest the features JJ and KK of the claimed invention. Therefore, Rao and Szeto, when combined cannot disclose or suggest the features R and S of the claimed invention. Nor is there any reason to combine and modify the teachings of these references in an attempt to arrive at the subject claims.

Alleged Combination of Rao + Szeto

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. Rao specifies named executables to be used when applying a firmware update.

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant respectfully disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, there is no reason why a person skilled in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto.

Nor is there a reason to do so. Furthermore, the combination of these documents does not disclose any feature of independent claim 22.

Accordingly, for at least the above reasons, independent claim 22 is believed to be new and non-obvious in view of the cited art. Similarly, claims 23 and 25-28, although containing their own allowable subject matter, also are considered to be new and non-obvious at least in view of their dependency from an allowable independent claim.

Claim 24

Independent claim 24 currently recites:

LL) A data structure embodied on a computer-readable medium, comprising:

MM) commands, execution of which create at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and

NN) a further command identifying the first leaf node and specifying execution of an unidentified executable on the first data stored at the first leaf node.

Rao

Appellant agrees with the Examiner that Rao does not disclose NN because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

Appellant respectfully disagrees with the Examiner that Rao discloses LL because, for example, there is no disclosure whatsoever of a data structure embodied on a computer-readable medium, for example, a smart card (page 7, lines 10 to 11).

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable.

Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

Szeto

Appellant respectfully disagrees with the Examiner that Szeto discloses NN.

Szeto does not disclose or suggest 'NN' because, for example:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the invention, the execution is commanded by the sending client.**
As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.
   - column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

   Contrastingly, embodiments of the invention use a data structure embodied on a computer-readable medium (i.e. a smart card as a sending client – page 7, lines 10 to 11) to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the invention, the executable is unidentified at the sending client.**
As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the invention, the data structure embodied on a computer-readable medium does not contain any information about the executable.

Szeto does not disclose or suggest LL because, for example there is no disclosure whatsoever of a data structure embodied on a computer-readable medium.

Szeto does not disclose or suggest MM because there is no disclosure whatsoever of commands to create a hierarchical data structure at an electronic device.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant

messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. smart card – page 7, lines 10 to 11) ability to specify execution in one or more mobile phone's using a non-specific command message. Therefore the control aspect of embodiments of the present invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur.

Combination Problems

Neither Rao nor Szeto individually disclose or suggest the features LL and NN of the claimed invention. Therefore Rao and Szeto, when combined cannot disclose or suggest the features LL and NN of the claimed invention. Nor is there any reason to combine and modify the teachings of these references in an attempt to arrive at the subject claims.

Alleged Combination of Rao + Szeto

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. Rao specifies named executables to be used when applying a firmware update.

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant respectfully disagrees with the permissibility of combining Rao and Szeto. It

is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, there is no reason why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Furthermore, the combination of these documents does not disclose or suggest the features of independent claim 24.

The combination Rao with Szeto is not obvious because Szeto is not relevant to SyncML technology and Rao is specific to SyncML technology.

Furthermore, Szeto is concerned with instant messaging communications, whereas Rao is concerned with firmware updating. The object to be attained in Rao is how to update firmware using SyncML. Szeto does not relate to firmware updates.

Thus, the combination of Rao with Szeto is considered impermissible, and is therefore non-obvious. One skilled in the art would not look to combine the teachings of these references. Nor would the combination disclose or suggest the claimed invention.

The Examiner, in his Advisory Action, comments that Szeto is "pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata". Szeto does not identify "the corresponding application" (supporting application) using metadata but instead identifies the supporting application using an identifier specified by a sending IM client, as explained above. Szeto therefore is not

relevant to the Examiner-defined "problem with which the applicant was concerned". Therefore the use of Szeto in combination with Rao would appear impermissible by the Examiner's and the Appellant's reckoning.

Furthermore, the "problem with which the applicant was concerned" is using a non-specific command sent from a device to perform a common process on a plurality of target devices without specific adaptation for each device (page 1, line 33 to page 2, line 3; page 3, lines 9 to 11, lines 13 to 14). In Szeto, the sending client sends an IM message to a receiving client, where an environment of the receiving client determines how the message is to be processed. Therefore, in Szeto, the receiving client decides whether and what to do with the identified data, whereas in the present invention the sending client tells the receiving client whether and what to do with the identified data. The receiving client of embodiments of the invention determines how to do it. Thus, Szeto teaches away from the claimed invention by specifying the control of received messages at the receiving client and not at the sending client.

Accordingly, for at least the foregoing reasons, independent claim 24 is believed to be new and non-obvious in view of the cited art.

Claim 35

Independent claim 35 currently recites:

OO) A method, comprising:
providing code identifying first data and specifying execution of an unidentified executable on the first data; and

PP) transmitting the code.

Rao

Appellant respectfully agrees with the Examiner that Rao does not disclose OO because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

Szeto

Appellant respectfully disagrees with the Examiner that Szeto discloses OO.

Szeto does not disclose or suggest 'OO' because, for example:

c) **In Szeto, the execution is specified by the receiving client, whereas in embodiments of the present invention, the execution is specified by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is specified by the IM environment of the receiving client and not the IM message, i.e.

- column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

Contrastingly, embodiments of the present invention use a code (i.e. provided by a sending client such as a smart card – page 7, lines 10 to 11), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

d) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the invention, the executable is unidentified at the sending client.**

As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the invention, the provided code does not contain any information about the executable.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the

receiving client using an environment.

Embodiments of the invention are directed to the sending client's ability to specify execution in one or more mobile phone's using a non-specific command message. Therefore the control aspect of embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur.

Combination Problems

Rao nor Szeto individually disclose or suggest, for example, the feature OO of the claimed invention. Therefore, Rao and Szeto, when combined cannot disclose or suggest he feature OO of the claimed invention. Nor is there any reason to combine and modify the teachings of these references in an attempt to arrive at the subject claims.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. Rao specifies named executables to be used when applying a firmware update.

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant respectfully disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified

from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, there is no reason why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Nor is there any reason to do so. Furthermore, the combination of these documents does not disclose independent claim 35.

Claim 36

Independent claim 36 recites:

QQ) A method, comprising:
transmitting commands for creating a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and

RR) transmitting a further command specifying execution of an unidentified executable on the first data stored at the first leaf [node].

Rao

Appellant agrees with the Examiner that Rao does not disclose RR because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the

features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

Szeto

Appellant disagrees with the Examiner that Szeto discloses RR.

Szeto does not disclose or suggest 'RR' because, for example:

e) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the invention, the execution is commanded by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.

- column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

Contrastingly, embodiments of the invention use commands transmitted by a service provider (i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

f) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the invention, the executable is unidentified at the sending client.** As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the invention, the transmitted command does not contain any information about the executable.

g) There is no disclosure whatsoever in Szeto of a first leaf node or first data stored at the first leaf node.

Szeto does not disclose or suggest QQ because, for example, there is no disclosure whatsoever of a hierarchical nodular data structure or commands for the creation of a hierarchical nodular data structure.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant

messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to create a hierarchical nodular data structure at one or more mobile phones and specify execution using a non-specific command message. Therefore the control aspect of the embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

Rao nor Szeto individually disclose or suggest, for example, the feature RR of the claimed invention. Therefore Rao and Szeto, when combined cannot disclose or suggest the feature RR of the claimed invention. Nor is there any reason to combine and modify the teachings of these references in attempt to arrive at the subject claims.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. Rao specifies named executables to be used when applying a firmware update.

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant disagrees with the permissibility of combining Rao and Szeto. It is not known

or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, there is no reason why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Nor is there any reason to do so. Furthermore, the combination of these documents does not disclose or suggest independent claim 36.

Accordingly, for at least the foregoing reasons, independent claim 36 is believed to be new and non-obvious in view of the cited art.

Claims 37 to 39

Independent claim 37 recites:

SS) A server, comprising:
a memory configured to store code identifying first data and specifying execution of an unidentified executable on the first data; and

TT) an interface configured to transmit the code.

Appellant agrees with the Examiner that Rao does not disclose SS because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the

rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

Szeto

Appellant respectfully disagrees with the Examiner that Szeto discloses SS.

Szeto does not disclose or suggest 'SS' because, for example:

h) **In Szeto, the execution is specified by the receiving client, whereas in embodiments of the invention, the execution is specified by the sending client.** As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is specified by the IM environment of the receiving client and not the IM message, i.e.

- column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

Contrastingly, embodiments of the invention use a command transmit from a server (i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).

i) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the invention, the executable is unidentified at the sending client.**

As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the invention, the server does not provide any information about the executable.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the receiving client using an environment.

Embodiments of the invention are directed to the sending client's ability to specify execution in one or more mobile phone's using a non-specific command message. Therefore, the control aspect of embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur.

Combination Problems

Rao nor Szeto individually disclose or suggest, for example, the feature SS of the claimed invention. Therefore, Rao and Szeto, when combined cannot disclose or suggest the feature SS of the claimed invention. Nor is there any reason to combine and modify the teachings of these references in an attempt to arrive at the subject claims.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. Rao specifies named executables to be used when applying a firmware update.

Szeto relates to controlling an application in a shared environment which exists between two or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant respectfully disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication

purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, there is no reason why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Nor is there any reason to do so. Furthermore, the combination of these documents does not disclose or suggest independent claim 37.

Accordingly, for at least the forgoing reasons, independent claim 37 is believed to be new and non-obvious in view of the cited art. Similarly, claims 38 and 39, although containing their own allowable subject matter, also are considered to be new and non-obvious at least in view of their dependency from an allowable independent claim.

Claim 40

Independent claim 40 recites:

UU) A server, comprising:
a memory configured to store commands, execution of which resulting in creation at an electronic device, of a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node, and configured to store a further command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first leaf node identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first leaf node; and

VV) a transmitter configured to transmit the stored instructions.

Rao

Appellant agree with the Examiner that Rao does not disclose UU because, for example, Rao uses commands that are executed by SyncML DM protocol (column 8, lines 65 to 67) and therefore specify a named executable to be used (i.e. the SyncML DM protocol uses exec commands to invoke enhancement commands for the firmware update).

It is respectfully asserted that the Examiner cannot properly rely on Rao's teaching in the rejection of Appellant's claims because Rao teaches the use of commands that are applied in a conventional way by specifying the use of a named executable. Rao does not disclose the features of the independent claims, nor does Rao teach towards the claimed invention. Rao teaches away from the claimed invention by specifying the use of a named executable. Therefore, Rao is not concerned with the claimed invention in any way apart from its relation to SyncML technology.

The Examiner is attempting to rely on the addition of Szeto to demonstrate obviousness of the claimed invention. Appellant respectfully asserts that any addition of Szeto to the teachings of Rao does not cure the shortcomings of Rao and does not disclose or suggest the claimed invention; nor is there any reason to combine and modify these references.

Szeto

Appellant respectfully disagrees with the Examiner that Szeto discloses UU.

Szeto does not disclose or suggest 'UU' because, for example:

a) **In Szeto, the execution is commanded by the receiving client, whereas in embodiments of the invention, the execution is commanded by the sending client.**
As illustrated in the above diagram at points 4 and 9, in Szeto, the execution of the data identified in the IM message is commanded by the IM environment of the receiving client and not the IM message, i.e.
   - column 12, line 49 to 53: "In step 1102, handlers in IM clients 202 receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step 1104".

   Contrastingly, embodiments of the present invention use commands transmitted by a server (i.e. sending client), to specify execution of identified data within a mobile phone (i.e. receiving client) (page 10, lines 11 to 12).
b) **In Szeto, the executable is identified by the sending client, whereas in embodiments of the invention, the executable is unidentified at the sending client.**

As illustrated in the above diagram at points 3, 7 and 8, the sending client in Szeto provides the receiving client with information about an IM application, and hence the supporting application (executable), in an IM message, i.e.

- column 3, lines 11 to 19: "... selecting, at a first client, the application in the instant messaging environment... including an identifier related to the application selected at the first client";
- abstract: "instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier [received in the IM message sent by the sending client], to execute the instant messaging application";

Contrastingly, in embodiments of the present invention, the transmitted command does not contain any information about the executable.

c) There is no disclosure whatsoever in Szeto of a hierarchical nodular data structure or commands for the creation of a hierarchical nodular data structure.

Effectively, a sending client in Szeto configures a message with information about a movie trailer. The message is sent to a receiving client. The receiving client has an 'environment' that decides what to do with the information in the received message. The environment of the receiving client may retrieve the identified movie trailer from a server using the information provided by the message. The environment also uses the information in the message to decide whether a 'supporting application' (i.e. a media player, content viewer, or other media-based display application – column 13, lines 7 to 10) is required to play the movie trailer. The environment then decides whether or not it wishes to play the movie trailer.

Environments at the receiving client are therefore used to control the processing of received messages. The environments decide whether and what to do with the received message and the information in contains.

The problem solved by Szeto is how to control the information received in an instant messaging environment. In particular, Szeto teaches the control of information at the

receiving client using an environment.

Embodiments of the invention are directed to the sending client's (e.g. service provider) ability to create a hierarchical nodular data structure at one or more mobile phones and specify execution using a non-specific command message. Therefore the control aspect of the embodiments of the invention is prominent at the sending client and not the receiving client.

Thus, Szeto teaches away from the claimed invention by using the receiving client to determine whether and what to do with received messages. Contrastingly, embodiments of the invention allow the sending client to control whether processing of a command message at the receiving client should occur and allow the receiving client to determine what executable should be used.

Combination Problems

Rao and Szeto individually do not disclose or suggest, for example, the feature UU of the claimed invention. Therefore, Rao and Szeto, when combined cannot disclose or suggest the feature UU of the claimed invention. Nor is there any reason to combine and modify these references in an attempt to arrive at the subject claims.

Rao relates to employing SyncML device management to facilitate firmware updates in an electronic device. Rao specifies named executables to be used when applying a firmware update.

Szeto relates to controlling an application in a shared environment which exists between 2 or more instant messaging users, for example, in the well-known MSN messenger. Messages used in the shared environment are directed at communication purposes and are addressed to specific IM clients. Environments are used to define how a received message at a receiving IM client should be processed.

The Appellant respectfully disagrees with the permissibility of combining Rao and Szeto. It is not known or specified from either document to use an instant messaging application, or

indeed an instant message to apply a firmware update. Equally, it is not known or specified from either document to use a firmware update for message communication between two IM clients. IM messages are clearly addressed to particular clients for messaging communication purposes and not to apply firmware updates.

Additionally, Rao specifically relates to SyncML technology using SyncML-specific "enhanced" commands. Therefore, there is no reason why a skilled person in the art would look to combine Rao with Szeto, when Szeto clearly does not relate to SyncML.

There is therefore no teaching, suggestion or motivation in Rao to combine it with Szeto. Nor is there any reason to do so. Furthermore, the combination of these documents does not disclose or suggest independent claim 40.

Accordingly, for at least the foregoing reasons, independent claim 40 is believed to be new and non-obvious in view of the cited art.

CONCLUSION

For at least the foregoing reasons, all claims are believed to be patentable. Moreover, it is respectfully noted again that although all dependent claims are believed to contain their own allowable subject matter, these claims should also be patentable for the additional reason that they depend from an allowable independent claim.

Accordingly, for at least the above reasons, the Appellant contends that the afore-cited references, whether viewed alone or in any combination, do not anticipate nor render obvious Appellant's claimed subject matter. The Appellant respectfully requests that the Board reverse the final rejection in the Final Office Action, and further that the Board rule that the pending claims are patentable over the cited art.

Respectfully submitted:
HARRINGTON & SMITH, PC

Christine Wilkes Beninati March 26, 2009

Christine Wilkes Beninati Date
Reg. No.: 37,967

Customer No.: 29683
HARRINGTON & SMITH, PC
4 Research Drive
Shelton, CT 06484-6212
Telephone: (203) 925-9400 ext. 17
Facsimile: (203) 944-0245
E-mail: cbeninati@hspatent.com

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Appeal Brief – Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Jessica Lee — 3.26.2009

Name of Person Making Deposit — Date

(8) CLAIMS APPENDIX

1. (Previously Presented) A method comprising:

receiving at an electronic device a command specifying execution of an unidentified executable on first data;

automatically determining, from metadata of the first data, a content type of the first data;

automatically identifying an executable using the content type determined from the metadata; and

operating on the first data using the identified executable.

2. (Cancelled)

3. (Previously Presented) A method as claimed in claim 1, wherein the command contains an identifier of the first data.

4. (Original) A method as claimed in claim 3, wherein the identifier identifies a node of a hierarchical nodular data structure.

5. (Original) A method as claimed in claim 4, wherein the command is an exec command and the identifier is a URI contained within a source element, which is contained within the *exec* command.

6. (Previously Presented) A method as claimed in claim 1, wherein the command is received as XML code.

7. (Original) A method as claimed in claim 6, wherein the command is a SyncML command.

8. (Previously Presented) A method as claimed in claim 1, wherein the identified first data is stored at the electronic device.

9. (Previously Presented) A method as claimed in claim 6, wherein the identified first data is

stored at a first leaf node of a hierarchical nodular data structure.

10. (Previously Presented) A method as claimed in claim 9, wherein the metadata is associated with the first leaf node and identifies the content type of the first data stored at the first leaf node of the hierarchical data structure.

11. (Cancelled)

12. (Previously Presented) A method as claimed in claim 1, wherein determining the content type uses at least one of the value of a Format element and the value of a Type element associated with the first data.

13. (Previously Presented) A method as claimed in claim 1 further comprising associating a plurality of different executables with each of a plurality of different content types.

14. (Previously Presented) A method as claimed in claim 13, wherein automatically identifying an executable from the content type comprises identifying the executable associated with the content type determined from the metadata.

15. (Previously Presented) A method as claimed in claim 13, wherein the plurality of different executables are stored in the electronic device.

16. (Previously Presented) A method as claimed in claim 1, further comprising, before receiving the command specifying execution of the unidentified executable on the first data, receiving commands for creating a hierarchical nodular data structure including the first data at the electronic device.

17. (Previously Presented) A method, comprising:
transferring code comprising a command to an electronic device, wherein the command specifies execution of an unidentified executable on first data stored at a first leaf node of a

hierarchical nodular data structure;

determining, from metadata of the first leaf node, a content type of the first data;

identifying an executable using the content type determined from the metadata of the identified first leaf node; and

operating on the first data, stored at the identified first leaf node, using the identified executable.

18. (Previously Presented) A method, comprising:

receiving re-usable code at an electronic device wherein the code comprises:

commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and a further command specifying execution of an unidentified executable on the first data stored at the first leaf node;

determining, from metadata stored at the first leaf node, a content type of the first data stored at the first leaf node;

identifying an executable using the content type determined from the metadata stored at the first leaf node; and

operating on the first data stored at the first leaf node using the identified executable.

19. (Previously Presented) An electronic device, comprising:

a memory configured to store first data and metadata of the first data;

a receiver configured to receive a command specifying execution of an unidentified executable on the first data; and

a processor configured to determine from the metadata of the first data, a content type of the first data, to identify an executable using the content type determined from the metadata, and to operate on the first data using the identified executable.

20. (Previously Presented) An electronic device as claimed in claim 19, wherein the receiver is further configured to receive ~~a~~ set-up code, and the processor is configured to interpret the received set-up code to create a hierarchical nodular data structure, having leaf nodes and interior

nodes, that comprises a first leaf node storing the first data.

21. (Previously Presented) An electronic device as claimed in claim 20, wherein the receiver is configured to receive the command in the set up code, and the processor is configured to interpret the command to determine, from the metadata of the first data, the content type of the first data.

22. (Previously Presented) A data structure embodied on a computer-readable medium, comprising:

code identifying first data and specifying execution of an unidentified executable on the first data.

23. (Previously Presented) A data structure as claimed in claim 22, wherein the code further specifies the transfer of the first data to an electronic device.

24. (Previously Presented) A data structure embodied on a computer-readable medium, comprising:

commands, execution of which create at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and

a further command identifying the first leaf node and specifying execution of an unidentified executable on the first data stored at the first leaf node.

25. (Previously Presented) A method, comprising: using a data structure as claimed in claim 22.

26. (Previously Presented) A method comprising: setting-up an electronic device using a data structure as claimed in claim 22.

27. (Previously Presented) A method comprising: re-using the data structure as claimed in

claim 22, to set-up different electronic devices.

28. (Previously Presented) A server for storing and transmitting the data structure as claimed in claim 22.

29.-33. (Cancelled)

34. (Previously Presented) An electronic device, comprising:

means for storing first data;

means for receiving a command specifying execution of an unidentified executable on the first data;

means for determining, from metadata, a content type of the identified first data;

means for identifying an executable using the content type determined from the metadata; and

means for operating on the identified data using the identified executable.

35. (Previously Presented) A method, comprising:

providing code identifying first data and specifying execution of an unidentified executable on the first data and

transmitting the code.

36. (Previously Presented) A method, comprising:

transmitting commands for creating a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and

transmitting a further command specifying execution of an unidentified executable on the first data stored at the first leaf.

37. (Previously Presented) A server, comprising:

a memory configured to store code identifying first data and specifying execution of an unidentified executable on the first data; and

an interface configured to transmit the code.

38. (Previously Presented) A server as claimed in claim 37, wherein the operations further comprise setting up an electronic device.

39. (Previously Presented) A server as claimed in claim 37, wherein the operations further comprise re-using the code in setting up different electronic devices.

40. (Previously Presented) A server, comprising:

a memory configured to store commands, execution of which resulting in creation at an electronic device, of a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node, and configured to store a further command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first leaf node; and

a transmitter configured to transmit the stored instructions.

41. (Previously Presented) A computer program product comprising program instructions embodied on a tangible computer-readable medium, execution of the program instructions resulting in operations comprising:

automatically determining, from metadata of first data, a content type of first data;

automatically identifying an executable using the content type determined from the metadata; and

enabling the first data to be operated on using the identified executable.

42. (Cancelled)

43. (Previously Presented) A method, comprising:

receiving a first command at an electronic device, the first command specifying creation of a leaf node in a hierarchical data structure, and identifying first data to be stored at the leaf node and metadata indicating a content type of the first data;

creating the leaf node at the electronic device;

receiving a second command, at the electronic device, that specifies execution of an unidentified executable on the first data stored at the created leaf node;

determining, from the metadata, the a content type of the first data;

identifying an executable using the content type determined from the metadata; and

operating on the first data using the identified executable.

44. (Previously Presented) An electronic device, comprising:

a receiver configured to receive a first command at an electronic device, the first command specifying creation of a leaf node in a hierarchical data structure, and identifying first data to be stored at the leaf node and metadata indicating a content type of the first data; and

a processor configured to create the leaf node at the electronic device, wherein

the receiver is further configured to receive a second command that specifies execution of an unidentified executable on the first data stored at the created leaf node, and the processor is further configured to determine, from the metadata, the content type of the first data, to identify an executable using the content type determined from the metadata, and to operate on the first data using the identified executable.

END OF CLAIMS

(9) EVIDENCE APPENDIX

The attached exhibits include copies of: the U.S. patent application for the application at issue (Exhibit A), the Final Office Action mailed on August 29, 2008 for the subject application (Exhibit B), and the three references relied on by the Examiner in rejecting the claims of the subject application (Exhibits C, D and E).

Exhibit A:	Subject US patent application serial number 10/589,155.
Exhibit B:	Final Office Action mailed on August 29, 2008 for the subject application.
Exhibit C:	Rao (US Patent 6,978,453).
Exhibit D:	DTD (SyncML Meta-Information DTD, version 1.0; pages 1-15.
Exhibit E:	Szeto (US Patent 7,188,143)
Exhibit F:	Advisory Action dated November 25, 2008

(10) RELATED PROCEEDINGS APPENDIX

There are no known decisions rendered by a court or the Board in any proceeding identified pursuant to paragraph (c)(1)(ii) of 37 C.F.R. §41.37.

EXHIBITS A-F

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 September 2005 (15.09.2005)

PCT

(10) International Publication Number
WO 2005/086511 A1

(51) International Patent Classification[7]: H04Q 7/32, G09G 5/00, G06F 15/167

(21) International Application Number: PCT/IB2004/000811

(22) International Filing Date: 14 February 2004 (14.02.2004)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **NOKIA CORPORATION** [FI/FI]; Keilalahdentie 4, FIN-02150 Espoo (FI).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **PEDERSEN, Claus** [DK/DK]; Nordmarksvænge 44, DK-2625 Vallensbæk (DK). **HANSEN, Jacob** [DK/DK]; Indre Vordingborgvej 18, DK-4700 Næstved (DK).

(74) Agent: **HIGGIN, Paul**; Swindell & Pearson, 48 Friar Gate, Derby DE1 1GY (GB).

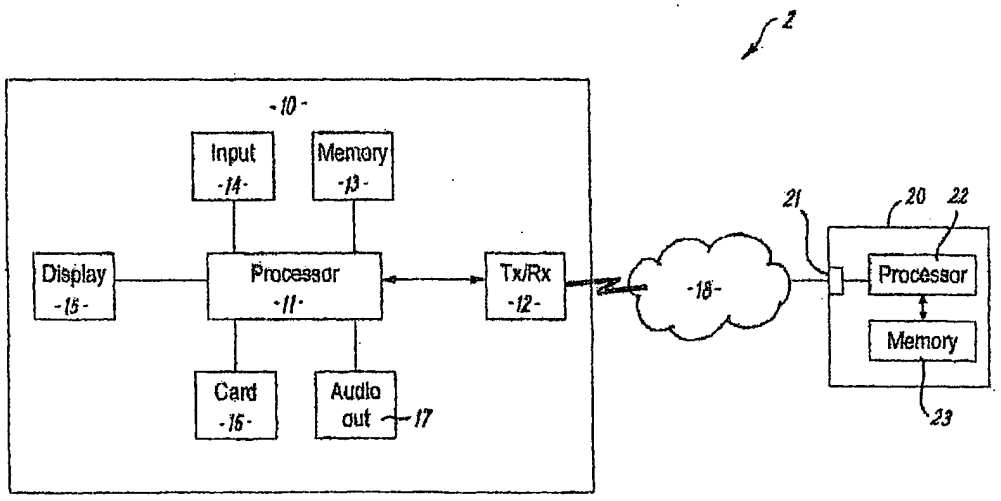
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A METHOD FOR CONFIGURING AN ELECTRONIC DEVICE



(57) Abstract: A method for automatically configuring an electronic device comprising: receiving at an electronic device a command identifying first data; automatically determining a property of the identified first data; automatically identifying an executable from the determined property; and operating on the identified first data using the identified executable.

WO 2005/086511 A1

TITLE

A method for configuring an electronic device.

FIELD OF THE INVENTION

Embodiments of the present invention relate to configuring an electronic device. Particular embodiments relate to a standard process for configuring mobile cellular telephones at, for example, the point of sale.

BACKGROUND OF THE INVENTION

A current trend is for electronic devices to have more features. This can make the electronic devices difficult to configure.

A user of a device may spend a considerable amount of time and effort configuring a device's settings so that it works correctly and/or as they would wish. A new device may appear less attractive to a user because the time and effort required may be very great.

This is particularly true for mobile cellular telephones. Mobile telephones are typically designed so that they can be configured for use in any one of a plurality of different cellular radio telephone networks controlled by different operators. However, each operator may require different settings to enable the telephone to operate correctly or in the way in which the operator desires. The operator (service provider) may wish to customise the mobile telephone with the necessary settings. In addition, a mobile telephone is personal device and a user may wish to configure it with their favorite ring tones, internet bookmarks, screen savers etc.

It is therefore desirable to provide some form of automatic set-up process for electronic devices and, in particular, mobile cellular telephones.

It would be desirable for this set-up process to be such that it can be used with

more than one electronic device without specific adaptation for that device. For example, it would be desirable if the set-up process may be used to initially configure all or most mobile cellular telephones.

The inventors considered whether automatic set-up would be possible using a SyncML message to transfer data to a target device and perform executables on the transferred data at the target device.

SyncML ™ Device Management (DM) is an open, universal industry standard. It gives third parties, such as service providers and corporate information management departments, the ability to create and manage a management tree stored in a mobile device. SyncML Device Management Tree and Description, v1.1.1, 10th Feb 2003, (www.syncml.org) describes the creation and maintenance of a management tree. A management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child (dependent) nodes but can store a value. A value may be a string, a file, a number etc. SyncML DM therefore provides a mechanism for providing data on-the-fly to the mobile device.

SyncML Representation Protocol, v1.1, 15th Feb 2002, (www.syncml.org) specifies the common Extensible Markup Language (XML) syntax and semantics usable by all SyncML protocols, including the Data Management (DM) protocol. The commands include: *Add*, *Copy*, *Delete*, *Exec*, *Get*, *Replace*. *Exec* allows the originator of the command to ask that a named executable is invoked by the recipient.

SyncML Device Management Tree and Description, v1.1.1, 10th Feb 2003, (www.syncml.org) describes how *Add* is used to create a management tree. SyncML DM does not specify how an executable can be performed on particular data using SyncML DM. The *exec* command contains an *item* element, which contains a *target* element, which contains a *LocURI* element. The *LocURI* element specifies only the location of the executable in the device. Therefore the *exec*

command does not allow the data to be used by the executable to be specified.

Furthermore the *exec* command requires the identification of a suitable executable in the target device. There must therefore be prior knowledge of the identity of the executable at a target device, which is specified in the *LocURI* element of the *exec* command. This is complex and inconvenient and prevents a single SyncML Message or Package being used with multiple mobile devices.

It would be desirable to create a SyncML code is suitable for performing a common process on a plurality of target devices that does not require specific adaptation for use with each device.

It would be desirable to instruct an executable to be performed on particular data using SyncML code that can be re-used for other devices.

BRIEF DESCRIPTION OF THE INVENTION

According to one embodiment of the invention there is provided a method for automatically configuring an electronic device comprising: receiving at an electronic device a command identifying first data; automatically determining a property of the identified first data; automatically identifying an executable from the determined property; and operating on the identified first data using the identified executable.

According to another embodiment of the invention there is provided a method for configuring a mobile cellular telephone comprising: transferring code comprising a command to a mobile cellular telephone, wherein the command identifies a first leaf node of a hierarchical nodular data structure; determining a property of the identified first leaf node; identifying an executable from the determined property; and operating on data stored at the identified first leaf node using the identified executable.

According to another embodiment of the invention there is provided a method for

configuring a plurality of mobile cellular telephones comprising:
transferring re-usable code to a mobile cellular telephone wherein the code comprises: commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and a first command identifying the first leaf node;
determining a property of the identified first leaf node;
identifying an executable from the determined property;
and operating on the first data stored at the first leaf node using the identified executable.

According to another embodiment of the invention there is provided a mobile cellular telephone arranged for automatic configuration comprising: means for storing first data; means for receiving a command identifying the first data; means for determining a property of the identified first data; means for identifying an executable from the determined property; and means for operating on the identified first data using the identified executable.

According to another embodiment of the invention there is provided a data structure for re-use in setting-up different mobile cellular telephones, comprising: code identifying first data and specifying execution of an unidentified executable on the first data.

According to another embodiment of the invention there is provided a data structure for re-use in setting-up different electronic devices, comprising: commands for creating at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node.

Embodiments of the invention may enable an executable resident on a target device to be executed without specifying the identity of that executable. As the identity of a resident executable may vary from device to device, this allows SyncML code to be re-used to perform a common process on a plurality of target

devices.

Embodiments of the invention may enable an executable resident on a target device to be used on specified data.

According to another embodiment of the invention there is provided a system for creating a data structure for re-use in setting-up different electronic devices, comprising: means for associating each one of a plurality of user friendly commands with different code portions, each of which includes one or more commands.

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the present invention reference will now be made by way of example only to the accompanying drawings in which:

Fig. 1 illustrates client-server system comprising a mobile device communicating with a server 20;

Fig 2 illustrates a management tree data structure;

Fig 3 is a signaling diagram for download of a set-up code; and

Fig 4 schematically illustrates the content of the set-up code 50.

DETAILED DESCRIPTION OF THE INVENTION

Fig 1 illustrates a client-server system 2 comprising a mobile device 10 communicating via a cellular radio telephone network 18 with a server 20. The mobile device 10, in this embodiment a mobile cellular telephone, comprises: a processor 11, a cellular radio transceiver 12, a memory 13, an input device 14 e.g. a keypad, a display 15, a smart card 16 and an audio output device 17.

The processor 11 controls the mobile telephone 10. It is connected to write to and read from the memory 13. It receives input data from the keypad 14 and provides output data to the display 15 and audio output device 17. It controls the cellular radio transceiver so that it can communicate in the cellular telephone network 18

which may be, for example, a GSM or WCDMA network. The processor is also connected to the smart card 16, which at least provides user identification information to the processor such as the user's telephone number or IMSI. The operation of the processor 11 is controlled by software stored in the memory 13 and loaded into the processor. In operation, the processor receives and transmits data via the transceiver 12 and writes and reads data from the memory 13.

It should be appreciated, that in other embodiments the mobile cellular telephone may not have a smart card 16 and/or may have multiple processors.

The server 20 comprises an input/output interface 21 connected to the cellular radio network 18 either directly or indirectly, a processor 22 and a memory 23. The server 20 is a SyncML DM server. It issues SyncML DM commands to the mobile telephone 10 via the input/output interface 21 and correctly interprets responses from the mobile telephone 10.

In the mobile telephone 10, processor 11 operates as a management client (MC) and can maintain a management tree data structure 100 in the memory 13. The MC correctly interprets SyncML DM commands received from the server, executes appropriate actions in the mobile telephone 10 and sends back relevant responses to the issuing management server via the transceiver 12.

A management tree is a hierarchical nodular data structure by which the management client interacts with the mobile telephone 10. The MC may store or retrieve values from the tree and manipulate the properties of the tree. The management tree has nodes connected by branches. Each node can be uniquely addressed by a URI. A node may be an interior node, which may have any number of child (dependent) nodes, but cannot store any value or a node may be a leaf node, which cannot have child (dependent) nodes but can store a value. A value may be a string, a file, a number etc.

The management tree can be manipulated by the MC. New nodes can be created and the values at certain leaf nodes can be changed. There is synchronous run

time access to the leaf nodes and interior nodes.

As will be described in more detail below, the MC in response to receiving set-up code from the server 20 creates an 'operator' management object as part of the automatic set-up process. As illustrated in Fig. 3, this 'operator' management object 102 is, in this example, a sub-tree depending from the root 104 of the management tree 100 and is used during the set-up process.

The automatic set-up process may, for example, be initiated by inserting a smart card 16 into the cellular mobile telephone 10. The smart card 16 contains the necessary information to bootstrap the automatic set-up process. The mobile device 10 sends a download initiation message 60 as illustrated in Fig 3 to the server 20.

The server 20 stores set-up code 50 in memory 23, which is usable with multiple devices without adaptation. The server 20 in response to the download initiation message initiates a SyncML Data Management (DM) Session 62. The DM session 62 is used to transfer the stored set-up code 50 to the mobile device 10.

The download initiation message 60 may be sent by any suitable means such as a Short Message Service (SMS) message or, if the device is a personal digital assistant without mobile telephone capabilities via IR, Bluetooth or a serial data connection such as USB.

The set-up code may be sent by any suitable means such as a Short Message Service (SMS) message or, if the device is a personal digital assistant without mobile telephone capabilities via IR, Bluetooth or a serial data connection such as USB.

The set-up code 50 is schematically illustrated in Fig. 4. The set-up code 50 is a data structure for re-use in setting-up different mobile devices.

The set-up code 50, in this example, comprises two portions, which are logically

separate but which may be interleaved. There is a first portion 52 for creating a management tree or updating an existing management tree. There is a second portion 54 for carrying out executables.

The first portion 52 for updating an existing management tree comprises a sub-portion for creating internal nodes of the management tree and a sub-portion for creating leaf nodes of the management tree.

As an example, the code may create an interior node 'Operator' 106 depending from the root 104 using XML code similar to this:

```
<Add>
<CmdID> 1</CmdID>
<Item>
        <Meta>
                <Format xmlns='syncml:metinf'> node   /<Format>
                <Type xmlns='syncml:metinf'>  interior /<Type>
        </Meta>
        <Target>
                <LocURI> /Operator  </LocURI>
        </Target>
</Item>
</Add>
```

As an example, the code may subsequently create an interior node 108 depending from the 'Operator' node 106 using XML code similar to this:

```
<Add>
<CmdID> 2</CmdID>
<Item>
        <Meta>
                <Format xmlns='syncml:metinf'> node   /<Format>
                <Type xmlns='syncml:metinf'>  interior /<Type>
```

```
        </Meta>
        <Target>
            <LocURI> /Operator/ring_tones </LocURI>
        </Target>
        </Item>
</Add>
```

As an example, the code may create a leaf node 110 depending from the node 108 using XML code similar to this:

```
<Add>
<CmdID> 3</CmdID>
<Item>
        <Meta>
            <Format xmlns='syncml:metinf'> format  /<Format>
            <Type xmlns='syncml:metinf'>  MIDI ringing tone /<Type>
        </Meta>
        <Target>
            <LocURI> /Operator/ring_tones/smashhit#1 </LocURI>
        </Target>
        <Data>  the data </Data>
</Item>
</Add>
```

where *the data,* is the data for creating the smashhit#1 ring tone in a format as defined by *format.*

The second portion 54 for carrying out executables comprises multiple sub-portions each of which is for carrying out an executable. The order of the sub-portions determines the order in which the executables are carried out. At least some of the sub-portions specify that an executable is carried out using particular data. As an example, the code for such a sub-portion may be XML code similar to this:

```
<Exec>
<CmdID> 3</CmdID>
<Item>
    <Source>
        <LocURI> /Operator/ring_tones/smashhit#1 </LocURI>
    </Source>
</Item>
</Add>
```

This *exec* command specifies execution of an unidentified executable on the data contained within 'source', that is the smashhit#1 ring tone. It should be noted that the *Exec* command does not specify which executable should be used and the meaning of the command depends upon the content type of the data which it identifies.

The DM client of the mobile cellular telephone processes the received set-up code 50. The portion 52 for updating an existing management tree is processed in accordance with the SyncML DM specifications. The DM client thus creates, according to the example given, a sub-tree 102 depending from the root as illustrated in Fig 2.

The portion 54 for carrying out executables is processed as follows. The code is parsed to identify the first sub-portion. The first sub-portion is parsed to identify the URI specified by the element *LocURI* contained within the element *source*. The element LocURI identifies a leaf node within the newly created sub-tree of the management tree.

The DM client accesses the identified leaf node, which in the example given above is Operator/ring_tones/smashhit#1. It reads the properties of the identified leaf node and in particular those properties contained within *meta*.

The DM client uses the content of the F*ormat* element and/or the content of the

Type to identify the content type of the data stored at the identified leaf node.

The DM client associates possible *Formats* and *Types* with different executables resident in the mobile telephone, for example using a look-up table. Thus, the DM client can associate an identified leaf node with an executable using the *Format* and/or *Type* of the leaf node. In this way, if the leaf node stores a sound file it is associated with an audio player, if the leaf node stores a video file it is associated with a video player, if the leaf node stores a picture file it is associated with a picture viewer, if the leaf node is a Java Midlet it is associated with the Java Virtual Machine (JVM), if the leaf node is contact details it is associated with an executable that adds it to the telephone's contact list, if the leaf node is a bookmark it is associated with an executable that adds the bookmark to the phone's bookmark list etc.

SyncML DM uses a number of different commands including:

Add: Allows an originator of the command to ask that a data element or data elements contained by the command are added to data accessible by the recipient. For example, when sent from a server to a mobile terminal it may add a node to a DM tree.

Copy: Allows an originator of the command to ask that a data element or data elements contained by the command that are accessible to the recipient are copied.

Delete: Allows the originator of the command to ask that a data element or data elements, contained in the command, that are accessible to the recipient are deleted. For example, when sent from a server to a mobile terminal it may remove a node from a DM tree.

Exec: Allows the originator of the command to ask that a named or supplied executable is invoked by the recipient.

Get: Allows the originator of the command to ask for a data element or elements from the recipient. For example, when sent from a mobile terminal to a server it may obtain the content of a node of the DM tree.

Replace: Allows the originator to ask that a data element or data elements accessible to the recipient be replaced.

SyncML DM also requires a special syntax to be use with each command. It would be desirable for each operator to be able to easily create suitable set-up code 50 without a detailed knowledge of the SyncML DM commands and their syntax. A computer program may be provided at the server 20 for doing this. The computer program effectively provides a macro that converts simple user friendly commands into the appropriate SyncML DM commands having the correct syntax.

The computer program may, for example, give the Operator the following options:

a) Install X
b) Preview X (with installation)
c) Preview X (without installation)
d) Execute X (installation implied)

The computer program, for example converts these options to:

a) a SyncML Add command
b) a SyncML Add command followed by a SyncML Exec command
c) a SyncML Add command followed by SyncML exec command followed by SyncML Delete command.
d) a SyncML Exec command.

Although embodiments of the present invention have been described in the preceding paragraphs with reference to various examples, it should be appreciated that modifications to the examples given can be made without departing from the scope of the invention as claimed. For example although described with reference to a mobile phone, it should be appreciated that the

present invention can find application in any user configurable electronic device. It has particular application in mobile devices such as mobile telephones and personal digital assistants, but may also find application in personal computers, for example.

Whilst endeavouring in the foregoing specification to draw attention to those features of the invention believed to be of particular importance it should be understood that the Applicant claims protection in respect of any patentable feature or combination of features hereinbefore referred to and/or shown in the drawings whether or not particular emphasis has been placed thereon.

I/we claim:

CLAIMS

1. A method for automatically configuring an electronic device comprising:
receiving at an electronic device a command identifying first data;
automatically determining a property of the identified first data;
automatically identifying an executable from the determined property; and
operating on the identified first data using the identified executable.

2. A method as claimed in claim 1, wherein the determined property of the identified data indicates a content type.

3. A method as claimed in claim 1 or 2, wherein the command contains an identifier of the first data.

4. A method as claimed in claim 3, wherein the identifier identifies a node of a hierarchical nodular data structure.

5. A method as claimed in claim 4, wherein the command is an exec command and the identifier is a URI contained within a source element, which is contained within the *exec* command.

6. A method as claimed in any preceding claim, wherein the command is received as XML code.

7. A method as claimed in claim 6, wherein the command is a SyncML command.

8. A method as claimed in any preceding claim, wherein the identified first data is stored at the mobile device.

9. A method as claimed in claim 6 , wherein the identified first data is stored as a first leaf node of a hierarchical nodular data structure.

10. A method as claimed in claim 9, wherein each leaf node of the hierarchical

nodular data structure has properties and the step of determining the content type uses the properties of the first leaf node.

11. A method as claimed in claim 9 or 10, wherein each leaf node of the hierarchical nodular data structure has metadata and the step of determining the content type uses the first leaf node's metadata.

12. A method as claimed in any preceding claim, wherein the step of determining the content type uses the value of a Format element and/or the value of a Type element associated with the first data.

13. A method as claimed in any preceding claim further comprising associating a plurality of different executables with each of a plurality of different properties.

14. A method as claimed in claim 11, wherein the step of automatically identifying an executable from the determined property comprises identifying the executable associated with the determined property.

15. A method as claimed in claim 13 or 14, wherein the plurality of different executables are stored in the electronic device.

16. A method as claimed in any preceding claim, further comprising, before receiving the command identifying the first data, receiving commands for creating a hierarchical nodular data structure including the first data at the electronic device.

17. A method for configuring a mobile cellular telephone comprising:
transferring code comprising a command to a mobile cellular telephone, wherein the command identifies a first leaf node of a hierarchical nodular data structure;
determining a property of the identified first leaf node;
identifying an executable from the determined property; and
operating on data stored at the identified first leaf node using the identified executable.

18. A method for configuring a plurality of mobile cellular telephones comprising:
transferring re-usable code to a mobile cellular telephone wherein the code comprises:
commands for creating at the electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and
a first command identifying the first leaf node;
determining a property of the identified first leaf node;
identifying an executable from the determined property; and
operating on the first data stored at the first leaf node using the identified executable.

19. A mobile cellular telephone arranged for automatic configuration comprising:
means for storing first data;
means for receiving a command identifying the first data;
means for determining a property of the identified first data;
means for identifying an executable from the determined property; and
means for operating on the identified data using the identified executable.

20. A mobile cellular telephone as claimed in claim 19, further comprising:
means for receiving set-up code; and
means for interpreting the received set-up code to create a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises a first leaf node storing the first data.

21. A mobile cellular telephone as claimed in claim 20, further comprising means for interpreting a first command within the received set-up code to determine a property of the leaf node identified by the first command.

22. A data structure for re-use in setting-up different mobile cellular telephones, comprising:
code identifying first data and specifying execution of an unidentified executable

on the first data.

23. A data structure as claimed in claim 22, wherein the code further specifies the transfer of the first data to the mobile device

24. A data structure for re-use in setting-up different electronic devices, comprising:

commands for creating at an electronic device a hierarchical nodular data structure, having leaf nodes and interior nodes, that comprises first data stored at a first leaf node; and

a first command identifying the first leaf node that specifies execution of an unidentified executable on the first data stored at the first node.

25. A server for storing and transmitting the data structure as claimed in claim 22, 23 or 24.

26. A system for creating a data structure for re-use in setting-up different electronic devices, comprising:

means for associating each one of a plurality of user friendly commands with different code portions, each of which includes one or more commands.

27. A system as claimed in claim 26, wherein a first user friendly command is associated with XML code comprising only a SyncML Add command .

28. A system as claimed in claim 26 or 27, wherein a second user friendly command is associated with XML code comprising a SyncML Add command followed by a SyncML Exec command.

29. A system as claimed in claim 26, 27 or 28, wherein a third user friendly command is associated with XML code comprising a SyncML Add command followed by a SyncML exec command followed by SyncML Delete command.

30. A method for automatically configuring an electronic device substantially as

hereinbefore described with reference to and/or as shown in the accompanying Figs.

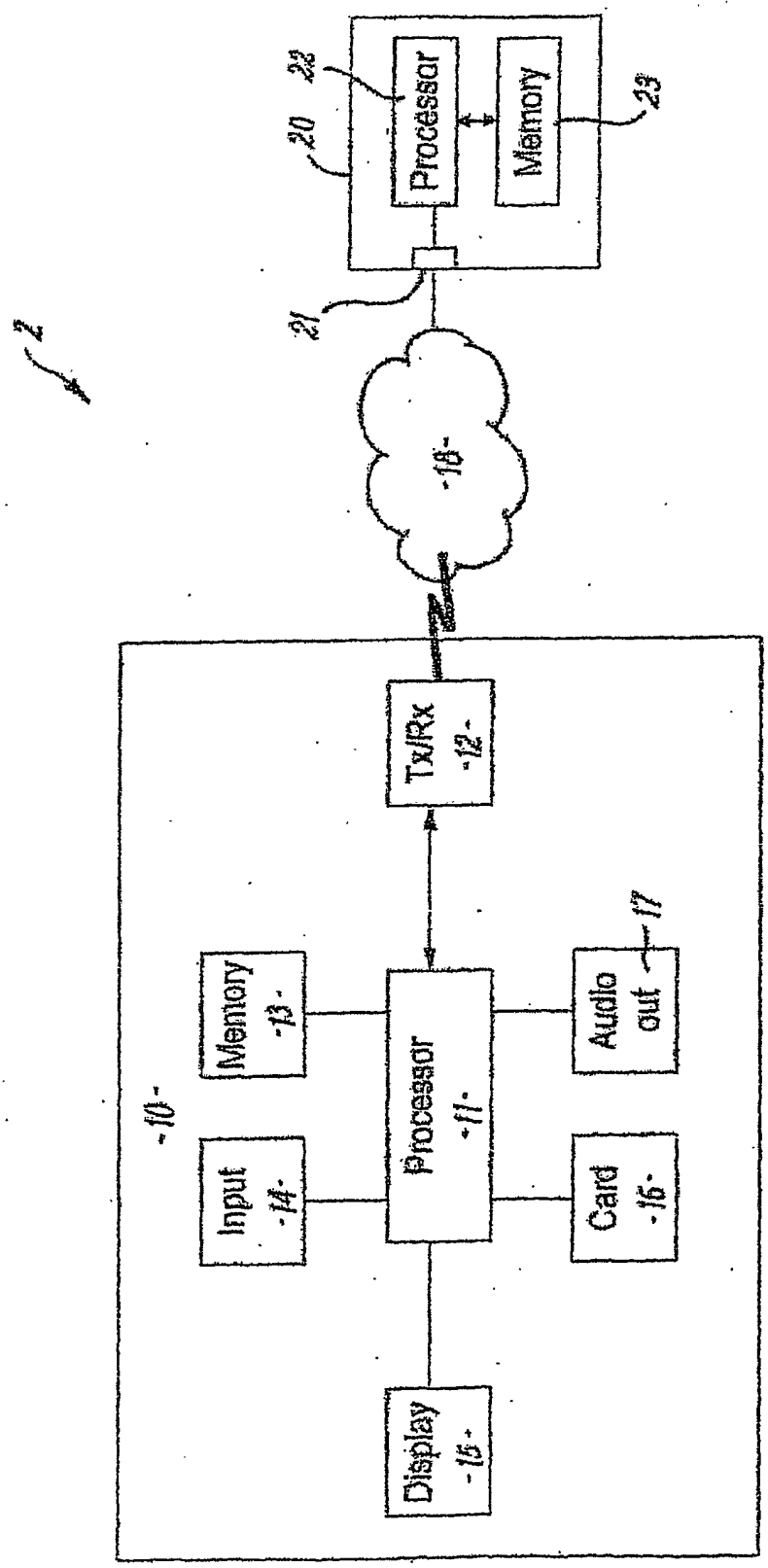

2
20
22
Processor
Memory
23
21
18
Tx/Rx
12
10
Memory
13
Input
14
Processor
11
Audio out
17
Card
16
Display
15

FIG. 1

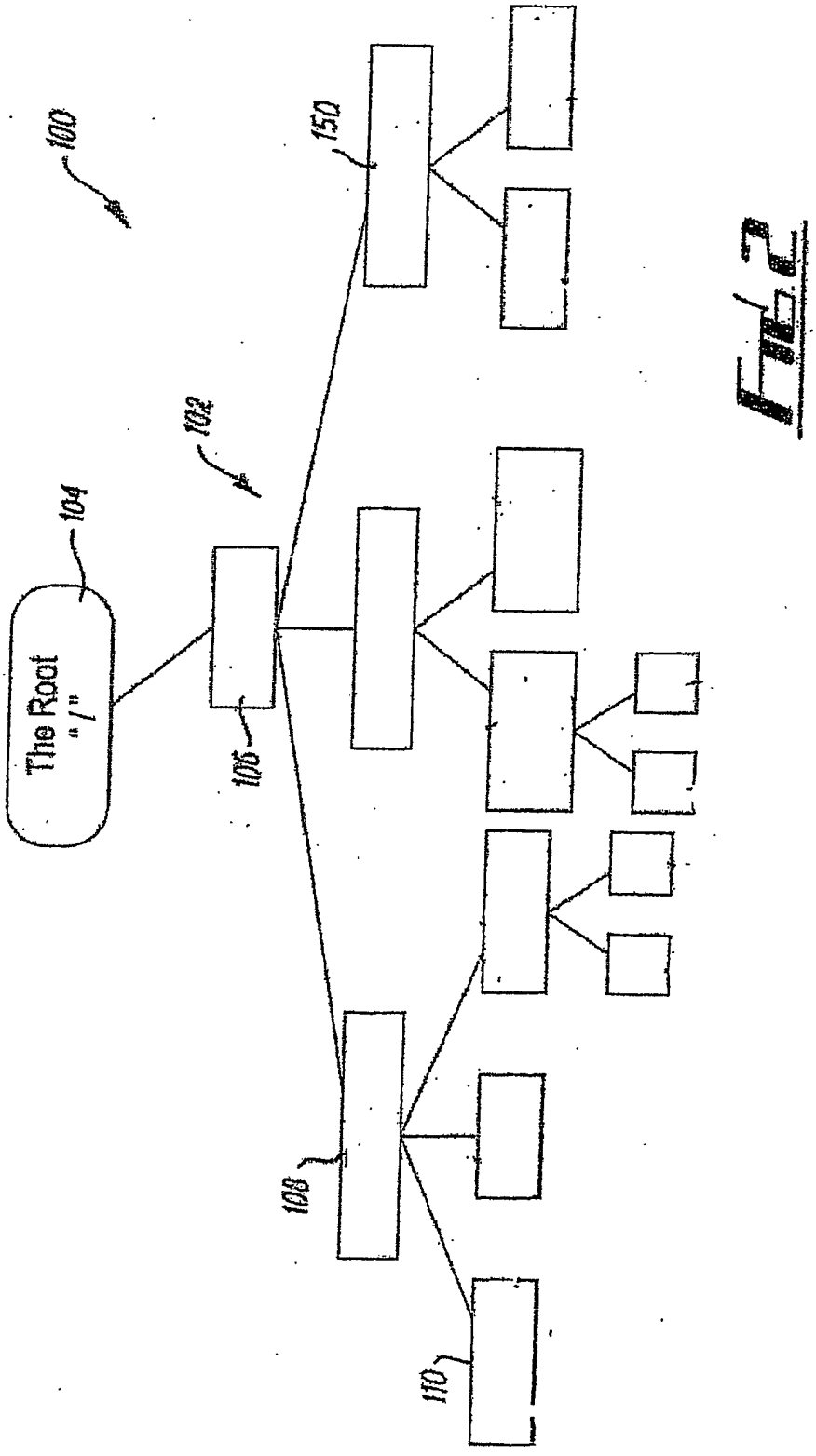


Fig. 2

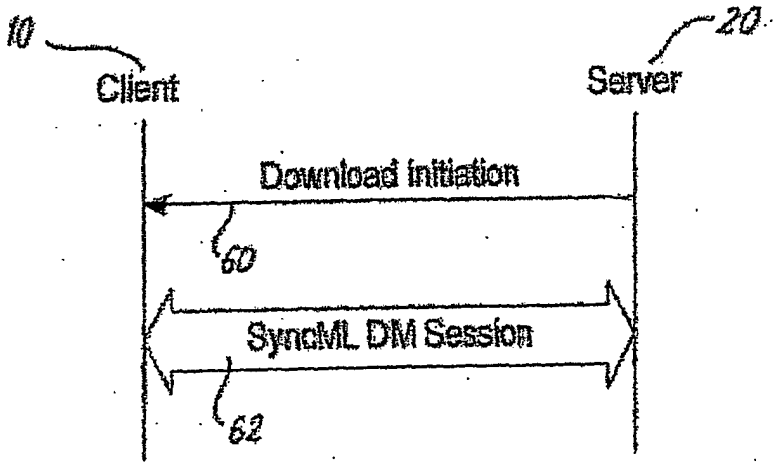

10
Client
20
Server
Download initiation
60
SyncML DM Session
62

FIG. 3

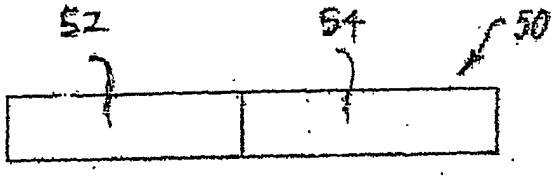

52
54
50

FIG. 4

1-ND/RO/CWB

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

B

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/589,155	05/04/2007	Claus Pedersen	884A.0148.U1(US)	6810

29683 7590 08/29/2008
HARRINGTON & SMITH, PC
4 RESEARCH DRIVE
SHELTON, CT 06484-6212

EXAMINER	
LEE, CHUN KUAN	
ART UNIT	PAPER NUMBER
2181	

MAIL DATE	DELIVERY MODE
08/29/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

2 mo.
3 mo.

USACTION
DUE DATE 10/29/08- 11/29/08-
PAPER DATED 8/29/08
OA FINAL 2/28/09
MSG PT DWG
APPEAL ISSUE FEE
OTHER

RECEIVED
SEP 2 2008
HARRINGTON & SMITH LLP

Office Action Summary	Application No.	Applicant(s)	
	10/589,155	PEDERSEN ET AL.	
	Examiner	Art Unit	
	Chun-Kuan Lee	2181	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 June 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-10,12-28,34-41,43 and 44* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-10,12-28,34-41,43 and 44* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *19 June 2008* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

RESPONSE TO ARGUMENTS

1. Applicant's arguments with respect to claims 1, 3-10, 12-21, 34 and 41 have been considered but are moot in view of the new ground(s) of rejection. Applicant's arguments filed for claims 22-28, 35-40 and 42 have been fully considered but they are not persuasive. Objection to the Drawings is withdrawn. Objection to the abstract is withdrawn. Rejection of claims 12 and 40 under 35 U.S.C.112 second paragraph is withdrawn. Currently, claims 2, 11, 29-33 and 42 are canceled, and claims 1, 3-10, 12-28, 34-41 and 43-44 are pending for examination.

2. In response to applicant's arguments (on page 14) with regard to the independent claim 1 rejected under 35 U.S.C. 103(a) that the Rao teaches "conventional" exe commands specified in the SyncML Representation Protocol; applicant's arguments have fully been considered, but are not found to be persuasive.

The examiner respectfully disagrees, as the examiner does not fully understand as to how the applicant concluded that Rao teaches "conventional" exe commands specified in the SyncML Representation Protocol because Rao's commands are "enhancement commands"; therefore, the commands are enhanced and are not conventional.

3. In response to applicant's arguments (on pages 14-16) regarding the independent claim 1 rejected under 35 U.S.C. 103(a) that the combination of references does not teach/suggest the claimed feature of receiving at an electronic device a command specifying execution of an unidentified executable; applicant's arguments have fully been considered, but are not found to be persuasive.

The examiner respectfully disagrees, as the examiner relied on the combination of references as following for the teaching of the above claimed feature:

Rao teaches receiving at an electronic device (Fig. 1, ref. 107) a command specifying execution of a data (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19), as the update command specifying execution of the firmware update data.

Szeto teach specifying execution of an unidentified executable (e.g. supporting application) on data (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the unidentified executable is specified base on the property of the data; more specifically, the received message do not specify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is specified.

4. In response to applicant's arguments (on pages 14-17) with regard to the independent claims 22, 24, 35-37 and 40 rejected under 35 U.S.C. 103(a) that the combination of references does not teach/suggest the claimed features determining a content type base on metadata, identifying using content type determined from

metadata, and identifying executable from metadata; applicant's arguments have fully been considered, but are not found to be persuasive.

Please note that the features upon which applicant relies (i.e., determining a content type base on metadata, identifying using content type determined from metadata, and identifying executable from metadata) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

I. OBJECTION TO THE CLAIMS

5. Claim 43 is objected to because of the following informalities:

in claim 43, line 3, "hierarchiacal" should be replace with -hierarchical-.

Appropriate correction is required.

II. REJECTIONS BASED ON 35 U.S.C. 112

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claims 16, 21, 34 and 43-44 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

As per claim 16, in line 2, it is not fully clear if "an unidentified executable" is the same/different unidentified executable previously recited; the examiner will assume the claimed limitation of "the unidentified executable" for the current examination.

As per claim 21, in line 4, it is not fully clear if "a content type" is the same/different content type previously recited; the examiner will assume the claimed limitation of "the content type" for the current examination.

As per claim 34, in line 4, it is not fully clear if "first data" is the same/different first data previously recited; the examiner will assume the claimed limitation of "the first data" for the current examination.

As per claim 43, in line 8, it is not fully clear if "a content type" is the same/different content type previously recited; the examiner will assume the claimed limitation of "the content type" for the current examination.

As per claim 44, in line 9, it is not fully clear if "a content type" is the same/different content type previously recited; the examiner will assume the claimed limitation of "the content type" for the current examination.

III. REJECTIONS BASED ON PRIOR ART

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-21, 34, 41 and 43-44 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rao et al. (US Patent 6,978,453) in view of "SyncML Meta-Information DTD" and Szeto (US Patent 7,188,143).

8. As per claim 1, Rao teaches a method comprising:

receiving at an electronic device (Fig. 1, ref. 107) a command (e.g. update command) specifying execution on first data (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19), wherein the first data executed to be associated with firmware update data;

utilization of metadata protocol, wherein the first leaf node would have the corresponding metadata (col. 6, l. 49 to col. 7, l. 19);

automatically determining (e.g. determining via recognition) a property of the identified first data (e.g. property identifying first data to be firmware update data) (col. 8, l. 25 to col. 12, l. 19), as the received command is recognized by the electronic device to have the property associated with firmware updating;

operating on the identified first data using an executable (e.g. module)(col. 5, ll. 23-32 and col. 5, l. 61 to col. 6, l. 4), as the module would operate on the firmware update data via downloading and updating processes.

Rao does not teach the method comprising: an unidentified executable; determine content type from the metadata; and automatically identifying an executable using the content type determined from the metadata.

SyncML Meta-Information DTD" teaches the metadata indicating a content type (Sec. 3-5 on pp. 5-12), as it is well known that metadata is data about data and SyncML have meta-information such as parameter or attributes that are about type or content of data; therefore, metadata may be utilized for determining the content type of data.

Szeto teach a system and method comprising:

an unidentified executable (e.g. unidentified supporting application) (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the received message do not identify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is identified;

determine a content type (e.g. application type) from metadata (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message having metadata with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation, and to determine the supporting application for the received message, the metadata is examining to determine the application type (e.g. content type); and

automatically identifying an executable (e.g. supporting application) using the content type determined from metadata (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message would be associated with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation in the metadata, and by using the content type, the corresponding supporting application would be identified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include SyncML Meta-Information DTD's content type and metadata and Szeto's identification of the executable into Rao's operation on the first identified data for the benefit of properly operating in accordance SyncML standard as in Rao's system and also for the benefit to the having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 1.

9. As per claim 3, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 1 as discussed above, where Rao further teaches the method comprising wherein the command contains an identifier (e.g. URI) of the first data (Rao, col. 6, l. 49 to col. 7, l. 19 and col. 8, ll. 25-34).

10. As per claim 4, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 3 as discussed above, where Rao further teaches the method comprising wherein the identifier identifies a node of a hierarchical nodular data structure (e.g. tree data structure) (Rao, col. 6, l. 49 to col. 7, l. 19 and col. 8, ll. 25-34).

11. As per claim 5, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 4 as discussed above, where Rao further teaches the method comprising wherein the command is an exec command and the identifier is a URI

contained within a source element, which is contained within the exec command (Rao, col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19).

12. As per claim 6, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 1 as discussed above, where Rao and Szeto further teach the method comprising wherein the command is received as XML code (Rao, col. 6, ll. 49 to col. 7, l. 3 and Szeto, col. 7, ll. 48-53).

13. As per claim 7, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 6 as discussed above, where Rao further teaches the method comprising wherein the command is a SyncML command (Rao, col. 6, ll. 49 to col. 7, l. 3 and col. 8, l. 25 to col. 12, l. 19).

14. As per claim 8, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 1 as discussed above, where Rao further teaches the method comprising wherein the identified first data is stored at the electronic device (Rao, col. 3, ll. 52-63; col. 5, ll. 23-32; col. 7, ll. 38-41 and col. 11, l. 48 to col. 12, l. 19).

15. As per claim 9, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 6 as discussed above, where Rao further teaches the method comprising wherein the identified first data is stored at a first leaf node of a hierarchical nodular data structure (e.g. tree data structure) (Rao, col. 3, ll. 52-63; col. 6, l. 49 to col.

7, l. 19; col. 8, ll. 25-34 and col. 11, l. 48 to col. 12, l. 19), as the data would be store in the first leaf node of the tree data structure.

16. As per claim 10, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 9 as discussed above, where all further teach the method comprising wherein the metadata is associated with the first leaf node and identifies the content type of the first data stored at the first leaf node of the hierarchical data structure (e.g. tree data structure) (Rao, col. 6, l. 49 to col. 7, l. 19; col. 8, l. 25 to col. 12, l. 19, SyncML Meta-Information DTD, pp. 5-6, and Szeto, Fig. 12A; col. 12, l. 66 to col. 13, l. 16).

17. As per claim 12, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 1 as discussed above, where SyncML Meta-Information DTD and Szeto further teach the method comprising wherein determining the content type uses at least one of the value of a Format element and the value of a Type element associated with the first data (SyncML Meta-Information DTD, pp. 5-12 and Szeto, Fig. 12A and col. 12, l. 66 to col. 13, l. 16).

18. As per claim 13, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 1 as discussed above, where Szeto further teaches the method comprising associating a plurality of different executables (e.g. different supporting applications for movie trailer, game, animation cartoon, advertisement, flash presentation) with each of a plurality of different content types (Szeto, Fig. 12A and col.

12, l. 66 to col. 13, l. 16), as each different content types have the corresponding supporting application.

19. As per claim 14, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 13 as discussed above, where SyncML Meta-Information DTD and Szeto further teach the method comprising wherein automatically identifying an executable from the content type comprises identifying the executable associated with the content type (SyncML Meta-Information DTD, pp. 5-12 and Szeto, Fig. 12A and col. 12, l. 66 to col. 13, l. 16).

20. As per claim 15, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 13 as discussed above, where Rao and Szeto further teach the method comprising wherein the plurality of different executables are stored in the electronic device (Rao, Fig 1; col. 5, l. 23 to col. 6, l. 4 and Szeto, Fig. 12A; col. 12, l. 66 to col. 13, l. 16), as the electronic devise would have the corresponding supporting application for operating the first data.

21. As per claim 16, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 1 as discussed above, where Rao further teaches the method comprising before receiving the command specifying execution of the unidentified executable on the first data, receiving commands for creating a hierarchical nodular data structure (e.g. tree data structure) including the first data at the electronic device

(Rao, col. 6, l. 49 to col. 7, l. 19 and col. 7, ll. 38-41 SyncML Meta-Information DTD, pp. 5-12), as the tree data structure is created prior to the execution of the update command.

22. As per claim 17, Rao teaches a method comprising:

transferring code comprising a command to an electronic device (Fig. 1, ref. 107), wherein the command specifies execution on first data stored at a first leaf node of a hierarchical nodular data structure (e.g. tree data structure) (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19), as the first leaf node would reside within the tree data structure;

utilization of metadata protocol, wherein the first leaf node would have the corresponding metadata (col. 6, l. 49 to col. 7, l. 19);

determining (e.g. determining via recognition) a property of the identified first data (e.g. property identifying first data to be firmware update data) (col. 8, l. 25 to col. 12, l. 19), as the received command is recognized by the electronic device to have the property associated with firmware updating; and

operating on the first data store at the identified first leaf node using an executable (e.g. module)(col. 5, ll. 23-32; col. 5, l. 61 to col. 6, l. 4 and col. 8, l. 25 to col. 12, l. 19), as the module would operate on the firmware update data via downloading and updating processes.

Rao does not teach the method comprising: an unidentified executable; determine a content type from the metadata; and identifying an executable using the content type determined from the metadata for operation.

SyncML Meta-Information DTD" teaches the metadata indicating a content type (Sec. 3-5 on pp. 5-12), as it is well known that metadata is data about data and SyncML have meta-information such as parameter or attributes that are about type or content of data; therefore, metadata may be utilized for determining the content type of data.

Szeto teach a system and method comprising:

an unidentified executable (e.g. unidentified supporting application) (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the received message do not identify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is identified;

determine a content type (e.g. application type) from metadata (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message having metadata with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation, and to determine the supporting application for the received message, the metadata is examining to determine the application type (e.g. content type); and

identifying an executable (e.g. supporting application) using the content type determined from metadata for operation (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message would be associated with content types including movie trailer, game, animation

cartoon, advertisement, and flash presentation in the metadata, and by using the content type, the corresponding supporting application would be identified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include SyncML Meta-Information DTD's content type and metadata and Szeto's identification of the executable into Rao's operation on the first identified data for the benefit of properly operating in accordance SyncML standard as in Rao's system and also for the benefit to the having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 17.

23. As per claim 18, Rao teaches a method, comprising:

receiving re-usable code at an electronic device (Fig. 1, ref. 107) (Fig. 1; col. 2, ll. 3-20; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19) (e.g. the code would be re-usable as SyncML specification enable operation with any mobile device) wherein the code comprises:

commands for creating at the electronic device a hierarchical nodular data structure (e.g. tree data structure), having leaf nodes and interior nodes (i.e. the tree data structure would have the corresponding leaf nodes and interior nodes), that comprises first data stored at a first leaf node; and a further command specifying execution on the first data stored at the first leaf node (e.g. executing firmware update data at first leaf node) (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19);

utilization of metadata protocol, wherein the first leaf node would have the corresponding metadata (col. 6, l. 49 to col. 7, l. 19);

determining (e.g. determining via recognition) a property of the first data stored at the first leaf node (col. 8, l. 25 to col. 12, l. 19), as the received command is recognized by the electronic device to have the property associated with firmware update data;

operating on the first data, stored at the first leaf node, using an executable (e.g. module) (col. 5, ll. 23-32 and col. 5, l. 61 to col. 6, l. 4), as the module would operate on the firmware update data via downloading and updating processes.

Rao does not teach the method comprising: an unidentified executable; determine a content type from the metadata; and identifying an executable using the content type determined from the metadata for operation.

SyncML Meta-Information DTD" teaches the metadata indicating a content type (Sec. 3-5 on pp. 5-12), as it is well known that metadata is data about data and SyncML have meta-information such as parameter or attributes that are about type or content of data; therefore, metadata may be utilized for determining the content type of data.

Szeto teach a system and method comprising:

an unidentified executable (e.g. unidentified supporting application) (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the received message do not identify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is identified;

determine a content type (e.g. application type) from metadata (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching,

the received message having metadata with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation, and to determine the supporting application for the received message, the metadata is examining to determine the application type (e.g. content type); and

identifying an executable (e.g. supporting application) using the content type determined from metadata for operation (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message would be associated with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation in the metadata, and by using the content type, the corresponding supporting application would be identified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include SyncML Meta-Information DTD's content type and metadata and Szeto's identification of the executable into Rao's operation on the first identified data for the benefit of properly operating in accordance SyncML standard as in Rao's system and also for the benefit to the having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 18.

24. As per claim 19, Rao teaches an electronic device, comprising:

a memory configure to store first data and metadata for first data (Fig. 1; col. 3, ll. 21-63; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19), as metadata protocol is utilized;

a receiver configured to receive a command specifying execution of the first data (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19), as the mobile device (Fig. 1, ref. 107) must have the receiver in order to receive commands from the SyncML server (Fig. 1, ref. 109) for execution of firmware updating; and

a processor (Fig. 1, ref. 107) configured to determine a property of the first data and operate on the data using an executable (col. 5, ll. 23-32; col. 5, l. 61 to col. 6, l. 4 and col. 8, l. 25 to col. 12, l. 19), as the received command is recognized by the electronic device to have the property associated with firmware updating, which the module would then operate on the firmware update data via downloading and updating processes.

Rao does not teach the method comprising: an unidentified executable; determine a content type from the metadata; and identifying an executable using the content type determined from the metadata for operation.

SyncML Meta-Information DTD" teaches the metadata indicating a content type (Sec. 3-5 on pp. 5-12), as it is well known that metadata is data about data and SyncML have meta-information such as parameter or attributes that are about type or content of data; therefore, metadata may be utilized for determining the content type of data.

Szeto teach a system and method comprising:

an unidentified executable (e.g. unidentified supporting application) (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the received message do not identify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is identified;

determine a content type (e.g. application type) from metadata (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message having metadata with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation, and to determine the supporting application for the received message, the metadata is examining to determine the application type (e.g. content type); and

identifying an executable (e.g. supporting application) using the content type determined from metadata for operation (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message would be associated with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation in the metadata, and by using the content type, the corresponding supporting application would be identified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include SyncML Meta-Information DTD's content type and metadata and Szeto's identification of the executable into Rao's operation on the first identified data for the benefit of properly operating in accordance SyncML standard as in Rao's system and also for the benefit to the having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 19.

25. As per claim 20, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 19 as discussed above, where Rao further teaches the electronic

device comprising wherein the receiver is further configured to receive a set-up code (e.g. set-up code such as add command), and the processor is operable to interpret the received set-up code to create a hierarchical nodular data structure (e.g. tree data structure), having leaf nodes and interior nodes, that comprises a first leaf node storing the first data (Rao, Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19), as the created tree data structure have the corresponding leaf nodes and interior nodes.

26. As per claim 21, Rao, SyncML Meta-Information DTD and Szeto teach all the limitations of claim 20 as discussed above, where all further teach the electronic device comprising wherein the receiver is configured to received the command in the set up code, and the processor is configured to interpret the command to determine, from the metadata of the first data, the content type (e.g. firmware update, movie trailer, game, animation cartoon, advertisement, flash presentation) of the first data (Rao, Fig. 1; col. 3, ll. 21-44; col. 5, ll. 23-32; col. 5, l. 61 to col. 6, l. 4; col. 6, l. 49 to col. 7, l. 19; col. 8, l. 25 to col. 12, l. 19; SyncML Meta-Information DTD, Sec. 3-5 on pp. 5-12, and Szeto, Fig. 12A; col. 12, l. 66 to col. 13, l. 16).

27. As per claim 34, Rao an electronic device (Fig. 1, ref. 107), comprising:

means for storing first data (Fig. 1; col. 3, ll. 52-63; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19);

means for receiving a command specifying execution on the first data (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19), wherein the first data is identified to be associated with firmware update data;

utilization of metadata protocol, wherein the first leaf node would have the corresponding metadata (col. 6, l. 49 to col. 7, l. 19);

means for determining (e.g. determining via recognition) a property of the identified first data (e.g. property identifying first data to be firmware update data) (col. 8, l. 25 to col. 12, l. 19), as the received command is recognized by the electronic device to have the property associated with firmware updating;

means for operating on the identified data using an executable (e.g. module)(col. 5, ll. 23-32 and col. 5, l. 61 to col. 6, l. 4), as the module would operate on the firmware update data via downloading and updating processes.

Rao does not teach the method comprising: an unidentified executable; determine a content type from the metadata; and means for identifying an executable using the content type determined from the metadata for operating.

SyncML Meta-Information DTD" teaches the metadata indicating a content type (Sec. 3-5 on pp. 5-12), as it is well known that metadata is data about data and SyncML have meta-information such as parameter or attributes that are about type or content of data; therefore, metadata may be utilized for determining the content type of data.

Szeto teach a system and method comprising:

an unidentified executable (e.g. unidentified supporting application) (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the received message do not identify the supporting

application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is identified;

determine a content type (e.g. application type) from metadata (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message having metadata with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation, and to determine the supporting application for the received message, the metadata is examining to determine the application type (e.g. content type); and

means for identifying an executable (e.g. supporting application) using the content type determined from metadata for operation (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message would be associated with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation in the metadata, and by using the content type, the corresponding supporting application would be identified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include SyncML Meta-Information DTD's content type and metadata and Szeto's identification of the executable into Rao's operation on the first identified data for the benefit of properly operating in accordance SyncML standard as in Rao's system and also for the benefit to the having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 34.

28. As per claim 41, Rao teaches a computer program product comprising program instructions embodied on a tangible computer readable-readable medium, execution of the program instructions resulting in operations comprising:

automatically determining (e.g. determining via recognition) a property of a first data (e.g. property identifying first data to be firmware update data) (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19), as the received command is recognized by the electronic device to have the property associated with firmware updating;

utilization of metadata protocol, wherein the first data would have the corresponding metadata (col. 6, l. 49 to col. 7, l. 19); and

enabling the first data to be operated on using an executable (e.g. module)(col. 5, ll. 23-32 and col. 5, l. 61 to col. 6, l. 4), as the module would operate on the firmware update data via downloading and updating processes.

Rao does not teach the computer program product comprising: determine a content type from the metadata; and automatically identifying an executable using the content type determined from the metadata for operation.

SyncML Meta-Information DTD" teaches the metadata indicating a content type (Sec. 3-5 on pp. 5-12), as it is well known that metadata is data about data and SyncML have meta-information such as parameter or attributes that are about type or content of data; therefore, metadata may be utilized for determining the content type of data.

Szeto teach a system and method comprising:

determine a content type (e.g. application type) from metadata (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message having metadata with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation, and to determine the supporting application for the received message, the metadata is examining to determine the application type (e.g. content type); and

automatic identifying an executable (e.g. supporting application) using the content type determined from metadata for operation (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message would be associated with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation in the metadata, and by using the content type, the corresponding supporting application would be identified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include Szeto's identification of the executable into Rao's operation of the first data for the benefit of having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 41.

29. As per claims 43-44, Rao teaches an electronic device and a method comprising:

a receiver configured to receive a first command at an electronic device (Fig. 1, ref. 107), the first command specifying creation of a leaf node in a hierarchical data structure (e.g. tree data structure with the corresponding leaf nodes), and identifying

first data (e.g. firmware updating data) to be stored at the leaf node and metadata of the first data (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 41 and col. 8, l. 25 to col. 12, l. 19), wherein the mobile device (Fig. 1, ref. 107) must have the receiver in order to receive command from the SyncML server (Fig. 1, ref. 109); and

a processor configured to create the leaf node at the electronic device (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 41 and col. 8, l. 25 to col. 12, l. 19), wherein

the receiver is further configured to receive a second command, at the electronic device, that specifies execution on the first data stored at the created leaf node, and the processor is further configured to operate on the first data using an executable (Fig. 1; col. 3, ll. 21-44; col. 5, l. 61 to col. 6, l. 4; col. 6, l. 49 to col. 7, l. 41 and col. 8, l. 25 to col. 12, l. 19).

Rao does not teach the electronic device and the method comprising: the metadata indicating a content type; an unidentified executable; determination of the content type from the metadata; and identify an executable using the content type determined from the metadata for operation.

SyncML Meta-Information DTD" teaches the metadata indicating a content type (Sec. 3-5 on pp. 5-12), as it is well known that metadata is data about data and SyncML have meta-information such as parameter or attributes that are about type or content of data; therefore, metadata may be utilized for determining the content type of data.

Szeto teach a system and method comprising:

an unidentified executable (e.g. unidentified supporting application) (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the received message do not identify the supporting

application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is identified;

determination of the content type (e.g. application type) from the metadata (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received message having metadata with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation, and to determine the supporting application for the received message, the metadata is examining to determine the application type (e.g. content type); and

identifying an executable (e.g. supporting application) using the content type determined from metadata for operation (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), in combination with SyncML Meta-Information DTD's teaching, the received metadata would be associated with content types including movie trailer, game, animation cartoon, advertisement, and flash presentation in the metadata, and by using the content type, the corresponding supporting application would be identified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include SyncML Meta-Information DTD's content type and metadata and Szeto's identification of the executable into Rao's operation on the first identified data for the benefit of properly operating in accordance SyncML standard as in Rao's system and also for the benefit to the having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claims 43-44.

30. Claims 22-28 and 35-40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rao et al. (US Patent 6,978,453) in view of Szeto (US Patent 7,188,143).

31. As per claim 22, Rao teaches a data structure embodied on a computer readable medium, comprising: code identifying first data (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19), as the first data associated with the firmware update data is identified; and

execution on the first data (col. 5, ll. 23-32 and col. 5, l. 61 to col. 6, l. 4), as the module would execute on the first data via downloading and updating processes.

Rao does not teach the data structure comprising specifying execution of an unidentified executable on the first data.

Szeto teach a system and method comprising specifying execution of an unidentified executable (e.g. supporting application) on a first data (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the unidentified executable is specified base on the property of the first data; more specifically, the received message do not specify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is specified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include Szeto's specification of the unidentified executable into Rao's execution of the first data for the benefit of having a reliable system and method

for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 22.

32. As per claim 23, Rao and Szeto teach all the limitations of claim 22 as discussed above, where Rao further teaches the data structure comprising wherein the code further specifies the transfer of the first data to an electronic device (Rao, Fig. 1, ref. 107) (Rao, Fig. 1; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19).

33. As per claim 24, Rao teaches a data structure embodied on a computer readable medium, comprising:

commands, execution of which create at an electronic device (Fig. 1, ref. 107) a hierarchical nodular data structure (e.g. tree data structure), having leaf nodes and interior nodes, that comprises first data stored at a first leaf node (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19), as the tree data structure would have the leaf nodes and interior nodes; and

a further command identifying the first leaf node that an executable (e.g. module) would operate on the first data stored at the first leaf node (col. 5, ll. 23-32; col. 5, l. 61 to col. 6, l. 4 and col. 8, l. 25 to col. 12, l. 19), as the module would operate on the firmware update data via downloading and updating processes.

Rao does not teach the data structure comprising specifying execution of an unidentified executable on the first data.

Szeto teach a system and method comprising specifying execution of an unidentified executable (e.g. supporting application) on a first data (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the unidentified executable is specified base on the property of the first data; more specifically, the received message do not specify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is specified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include Szeto's specification of the unidentified executable into Rao's execution of the first data for the benefit of having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 24.

34. As per claim 25, Rao and Szeto teach all the limitations of claim 22 as discussed above, where both further teach a method, comprising: using a data structure as claimed in claim 22 (Rao, Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19 and Szeto, Fig. 12A and col. 12, l. 66 to col. 13, l. 16).

35. As per claim 26, Rao and Szeto teach all the limitations of claim 22 as discussed above, where both further teach a method comprising: setting-up an electronic device (Rao, Fig. 1, ref. 107) using a data structure as claimed in claim 22 (Rao, Fig. 1; col. 3,

ll. 21-44; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19 and Szeto, Fig. 12A and col. 12, l. 66 to col. 13, l. 16).

36. As per claim 27, Rao and Szeto teach all the limitations of claim 22 as discussed above, where both further teach a method comprising: re-using the data structure as claimed in claim 22, to set-up different electronic devices (Rao, Fig. 1; col. 2, ll. 3-20; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19 and Szeto, Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the SyncML specification is able to work with any mobile device, therefore, the data structure would be re-usable.

37. As per claim 28, Rao and Szeto teach all the limitations of claim 22 as discussed above, where Rao further teaches a server (Rao, Fig. 1, ref. 109) for storing and transmitting the data structure as claimed in claim 22 (Rao, Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19).

38. As per claim 35, Rao teaches a method, comprising: providing code identifying first data (e.g. firmware update data); and transmitting the code (e.g. transmit to the mobile handset 107 of Fig 1) (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19 and col. 8, l. 25 to col. 12, l. 19).

Rao does not teach the method comprising specifying execution of an unidentified executable on the first data.

Szeto teach a system and method comprising specifying execution of an unidentified executable (e.g. supporting application) on a first data (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the specification of the supporting application is in accordance with the property of the first data; more specifically, the received message do not specify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is specified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include Szeto's specification of the unidentified executable into Rao's operation of the first data for the benefit of having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 35.

39. As per claim 36, Rao teaches a method, comprising:

transmitting commands for creating a hierarchical nodular data structure (e.g. tree data structure), having leaf nodes and interior nodes, that comprises first data stored at a first leaf node (Fig. 1; col. 3, ll. 21-44; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19), as the tree data structure would have the leaf nodes and interior nodes; and

transmitting a further command specifying execution on the first data stored at the first leaf node (col. 5, ll. 23-32; col. 5, l. 61 to col. 6, l. 4 and col. 8, l. 25 to col. 12, l.

19), as the module would operate on the firmware update data via downloading and updating processes.

Rao does not teach the method comprising specifying execution of an unidentified executable on the first data.

Szeto teach a system and method comprising specifying execution of an unidentified executable (e.g. supporting application) on a first data (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the specification of the supporting application is in accordance with the property of the first data; more specifically, the received message do not specify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is specified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include Szeto's specification of the unidentified executable into Rao's operation of the first data for the benefit of having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 36.

40. As per claim 37, Rao teaches a server (Fig. 1, ref. 109), comprising:

a memory configured to store a code identifying first data and operating on the first data; and an interface configure to transmit the code (Fig. 1; col. 3, ll. 21-44; col. 4, l. 62 to col. 5, l. 8; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19), wherein the interface may includes the SyncML engine (Fig. 1, ref. 137).

Rao does not teach the server comprising specifying execution of an unidentified executable on the first data.

Szeto teach a system and method comprising specifying execution of an unidentified executable (e.g. supporting application) on a first data (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the specification of the supporting application is in accordance with the property of the first data; more specifically, the received message do not specify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is specified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include Szeto's specification of the unidentified executable into Rao's operation of the first data for the benefit of having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 37.

41. As per claim 38, Rao and Szeto teach all the limitations of claim 37 as discussed above, where Rao further teaches a server comprising wherein the operations further comprise setting up an electronic device (Rao, Fig. 1, ref. 107) (Rao, Fig. 1; col. 3, ll. 21-44; l. 8; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19), as the firmware update data is utilized for setting up the mobile handset.

42. As per claim 39, Rao and Szeto teach all the limitations of claim 37 as discussed above, where Rao further teaches a server comprising wherein the operations further comprise re-using the code in setting up different electronic devices (Rao, Fig. 1; col. 2, ll. 3-20; col. 3, ll. 21-44; l. 8; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19), as the SyncML specification is able to work any mobile device, the code would be re-usable for different electronic device.

43. As per claim 40, Rao teaches a server (Fig. 1, ref. 109), comprising:

a memory configure to store commands, execution of which resulting in creation at an electronic device (Fig. 1, ref. 107), of a hierarchical nodular data structure (e.g. tree data structure), having leaf nodes and interior nodes, that comprises first data stored at a first leaf node, and configured to store a further command identifying the first leaf node for operating on the first data stored at the first leaf node (Fig. 1; col. 3, ll. 21-44; col. 4, l. 62 to col. 5, l. 8; col. 5, l. 23 to col. 6, l. 4; col. 6, l. 49 to col. 7, l. 19; col. 7, ll. 38-41 and col. 8, l. 25 to col. 12, l. 19), as the tree data structure includes the leaf nodes and interior nodes and the operation includes downloading and updating processes of update firmware data; and

a transmitter configured to transmit the stored instructions (Fig. 1; col. 3, ll. 21-44; col. 4, l. 62 to col. 5, l. 8 and col. 8, l. 25 to col. 12, l. 19), as the transmitter is needed in order to transfer the command to the mobile handset (Fig. 1, ref. 109).

Rao does not teach the server comprising specifying execution of an unidentified executable on the first data.

Szeto teach a system and method comprising specifying execution of an unidentified executable (e.g. supporting application) on a first data (Fig. 12A and col. 12, l. 66 to col. 13, l. 16), as the specification of the supporting application is in accordance with the property of the first data; more specifically, the received message do not specify the supporting application to be utilized with the message, and after the application type is determined from the received message, the corresponding supporting application is specified.

It would have been obvious for one of ordinary skill in this art, at the time of invention was made to include Szeto's specification of the unidentified executable into Rao's operation of the first data for the benefit of having a reliable system and method for a user to execute and control application (Szeto, col. 2, ll. 30-33) to obtain the invention as specified in claim 40.

IV. CLOSING COMMENTS

Conclusion

a. STATUS OF CLAIMS IN THE APPLICATION

The following is a summary of the treatment and status of all claims in the application as recommended by **M.P.E.P. 707.07(i)**:

a(1) CLAIMS REJECTED IN THE APPLICATION

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

b. DIRECTION OF FUTURE CORRESPONDENCES

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chun-Kuan (Mike) Lee whose telephone number is (571) 272-0671. The examiner can normally be reached on 8AM to 5PM.

IMPORTANT NOTE

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Alford Kindred can be reached on (571) 272-4037. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/C.K.L./

August 22, 2008

Chun-Kuan (Mike) Lee
Examiner
Art Unit 2181

/Alford W. Kindred/

Supervisory Patent Examiner, Art Unit 2181

Notice of References Cited	Application/Control No. 10/589,155	Applicant(s)/Patent Under Reexamination PEDERSEN ET AL.	
	Examiner Chun-Kuan Lee	Art Unit 2181	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	"SyncML Meta-Information DTD," 12/07/2000, Version 1.0, pages 1-15
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

US006978453B2

(12) **United States Patent**
Rao et al.

(10) **Patent No.: US 6,978,453 B2**
(45) **Date of Patent: Dec. 20, 2005**

(54) **SYSTEM WITH REQUIRED ENHANCEMENTS TO SYNCML DM ENVIRONMENT TO SUPPORT FIRMWARE UPDATES**

(75) Inventors: **Bindu Rama Rao**, Laguna Niguel, CA (US); **Patrick C. Lilley**, Irvine, CA (US)

(73) Assignee: **Bitfone Corporation**, Laguna Niguel, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 56 days.

(21) Appl. No.: **10/689,309**

(22) Filed: **Oct. 20, 2003**

(65) **Prior Publication Data**

US 2004/0083472 A1 Apr. 29, 2004

Related U.S. Application Data

(60) Provisional application No. 60/419,903, filed on Oct. 21, 2002.

(51) **Int. Cl.**[7] **G06F 9/44**
(52) **U.S. Cl.** **717/171**; 717/114; 717/139
(58) **Field of Search** 717/171, 114

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,261,055	A	11/1993	Moran et al.	395/275
5,442,771	A	8/1995	Filepp et al.	395/650
5,479,637	A	12/1995	Lisimaque et al.	395/430
5,579,522	A	11/1996	Christeson et al.	395/652
5,596,738	A	1/1997	Pope	395/430
5,598,534	A	1/1997	Haas	395/200.09
5,608,910	A	3/1997	Shimakura	395/670
5,623,604	A	4/1997	Russell et al.	395/200.1
5,666,293	A	9/1997	Metz et al.	395/200.5
5,752,039	A	5/1998	Tanimura	395/712
5,778,440	A	7/1998	Yiu et al.	711/154
5,790,974	A	8/1998	Tognazzini	701/204
5,878,256	A	3/1999	Bealkowski et al.	395/652
5,960,445	A	9/1999	Tamori et al.	707/203
6,009,497	A	12/1999	Wells et al.	711/103

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2339923	3/2000

(Continued)

OTHER PUBLICATIONS

P. Ciancarini et al, "Using a Coordination Language to Specify and Analyze Systems Containing Modile Components", ACM Transactions, vol. 9, No. 2 Apr. 2000, pp. 167-198.*

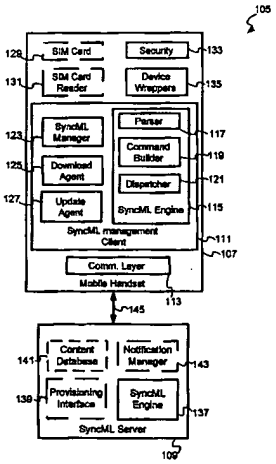
(Continued)

Primary Examiner—Todd Ingberg
(74) *Attorney, Agent, or Firm*—McAndrews, Held & Malloy, Ltd.

(57) **ABSTRACT**

A system for employing SyncML DM for updating firmware in mobile handsets and other devices. The system employs enhancements to SyncML DM specifications. A SyncML management client employs new commands, specified by the present invention, for retrieving update packages for firmware updates, for the verification of a received update package, the command for saving the update package in an appropriate management object, the command for initiating an update process by an update agent and the command for the subsequent notification of the results of processing by the update agent (success, failure, etc.). More specifically, the SyncML DM management client employs new commands, specified by the present invention, for retrieving update packages for firmware updates and for updating the firmware selectively based on appropriateness, security and authentication, employing fault tolerant means.

26 Claims, 2 Drawing Sheets



U.S. PATENT DOCUMENTS

6,038,636	A		3/2000	Brown, III et al.	711/103
6,064,814	A		5/2000	Capriles et al.	395/701
6,073,206	A		6/2000	Piwonka et al.	711/102
6,073,214	A		6/2000	Fawcett	711/133
6,088,759	A		7/2000	Hasbun et al.	711/103
6,105,063	A		8/2000	Hayes, Jr.	709/223
6,112,024	A		8/2000	Almond et al.	395/703
6,112,197	A		8/2000	Chatterjee et al.	707/3
6,126,327	A		10/2000	Bi et al.	395/200.51
6,128,695	A		10/2000	Estakhri et al.	711/103
6,157,559	A		12/2000	Yoo	365/52
6,163,274	A		12/2000	Lindgren	340/825.44
6,167,567	A	*	12/2000	Chiles et al.	717/173
6,198,946	B1		3/2001	Shin et al.	455/561
H001964	H	*	6/2001	Hoffpauir et al.	370/419
6,266,809	B1	*	7/2001	Craig et al.	717/173
6,279,153	B1		8/2001	Bi et al.	717/11
6,311,322	B1		10/2001	Ikeda et al.	717/1
6,438,585	B2		8/2002	Mousseau et al.	709/206
6,587,685	B2	*	7/2003	Mittal et al.	455/419
2001/0029178	A1		10/2001	Criss et al.	455/419
2001/0047363	A1		11/2001	Peng	707/104.1
2001/0048723	A1		12/2001	Peng	375/354
2002/0078209	A1		6/2002	Peng	709/227
2002/0116261	A1		8/2002	Moskowitz et al.	705/14
2002/0131404	A1		9/2002	Mehta et al.	370/352
2002/0152005	A1		10/2002	Bagnordi	700/234
2002/0156863	A1		10/2002	Peng	709/217
2002/0157090	A1		10/2002	Anton, Jr.	717/178
2003/0033599	A1		2/2003	Rajaram et al.	717/173
2003/0037075	A1		2/2003	Hannigan et al.	707/500
2003/0061384	A1		3/2003	Nakatani	709/245

FOREIGN PATENT DOCUMENTS

JP	8202626	8/1996
KR	2002-0034228	5/2000
KR	2001-0100328	11/2001

OTHER PUBLICATIONS

Mobile Streaming Media CDN Enabled by Dynamic SMIL, Takeshi Yoshimura et al, ACM, May 2002, pp. 651-661.*

* cited by examiner

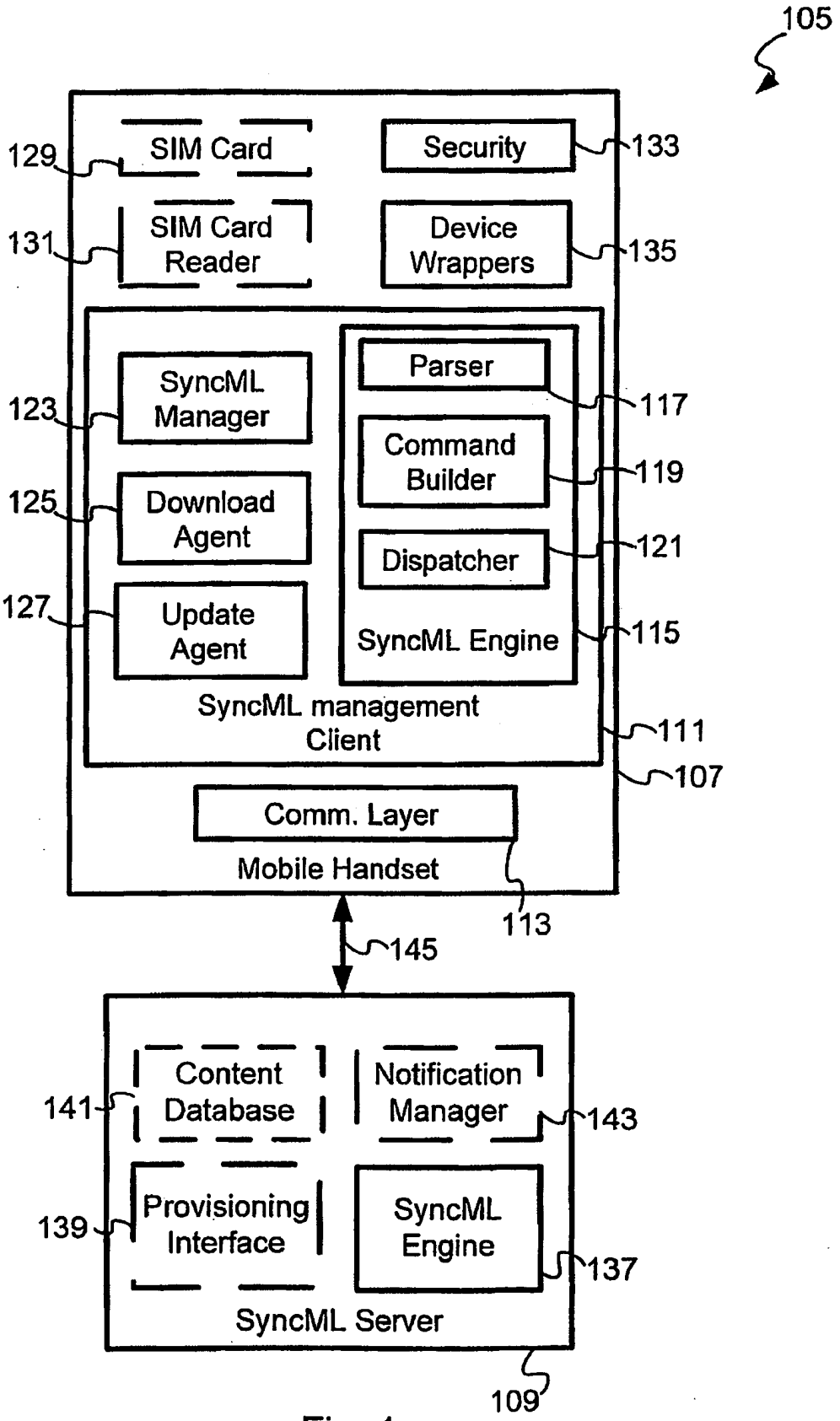

105
129
SIM Card
Security
133
131
SIM Card Reader
Device Wrappers
135
SyncML Manager
123
Parser
117
Command Builder
119
125
Download Agent
Dispatcher
121
127
Update Agent
SyncML Engine
115
SyncML management Client
111
107
Comm. Layer
Mobile Handset
113
145
Content Database
141
Notification Manager
143
139
Provisioning Interface
SyncML Engine
137
SyncML Server
109

Fig. 1

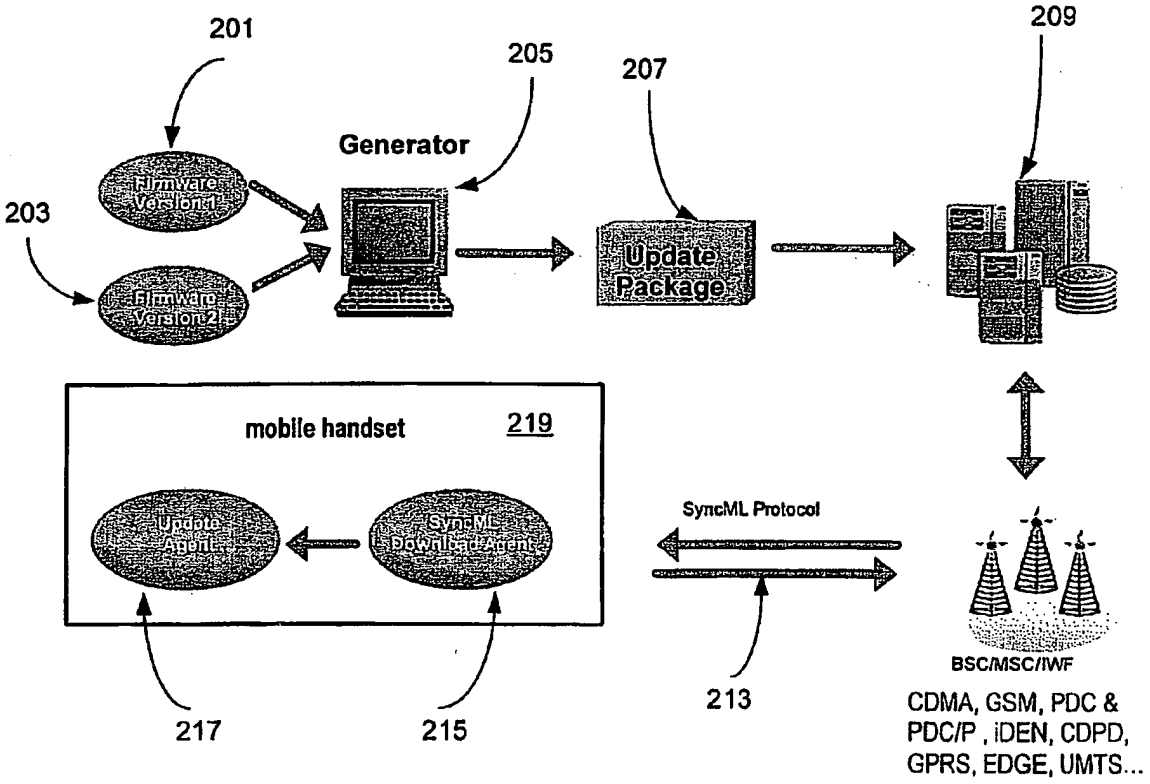

201
205
207
209
Generator
203
Firmware Version 1
Firmware Version 2
Update Package
mobile handset
219
Update Agent
SyncML Download Agent
SyncML Protocol
BSC/MSC/IWF
CDMA, GSM, PDC & PDC/P , iDEN, CDPD, GPRS, EDGE, UMTS...
217
215
213

Fig. 2

SYSTEM WITH REQUIRED ENHANCEMENTS TO SYNCML DM ENVIRONMENT TO SUPPORT FIRMWARE UPDATES

RELATED APPLICATIONS

This patent application makes reference to, claims priority to and claims benefit from U.S. Provisional Patent Application Ser. No. 60/419,903, entitled "System with Required Enhancements to SyncML DM Environment to Support Firmware Updates," filed on Oct. 21, 2002.

The complete subject matter of the above-referenced United States Patent Application is hereby incorporated herein by reference, in its entirety. In addition, this application makes reference to U.S. Provisional Patent Application Ser. No. 60/249,606, entitled "System and Method for Updating and Distributing Information", filed Nov. 17, 2000, and International Patent Application Publication No. WO 02/41147 A1, entitled "Systems And Methods For Updating And Distributing Information," publication date Mar. 23, 2002, the complete subject matter of each of which is hereby incorporated herein by reference, in its entirety.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[Not Applicable]

MICROFICHE/COPYRIGHT REFERENCE

[Not Applicable]

FIELD OF THE INVENTION

The present invention relates generally to the generation and dissemination of update packages for software and/or firmware employing the Synchronization Markup Language (SyncML) device management (DM) specifications and, more specifically, to the enhancements to SyncML DM in terms of new commands that facilitate firmware updates from one version to another.

BACKGROUND OF THE INVENTION

Electronic devices, such as mobile phones and personal digital assistants (PDAs), often contain firmware and application software either provided by the manufacturer of the electronic devices, by telecommunication carriers, or by third parties. The firmware and application software often contain software bugs. New versions of the firmware and software are periodically released to fix the bugs or to introduce new features, or both.

SyncML, founded in February 2000, is an open industry standard for a common data synchronization protocol that facilitates the transport of network data and personal information across multiple networks, platforms and devices. Ericsson, IBM, Lotus, Matsushita, Motorola, Nokia, Starfish Software, Symbian, and Openwave sponsor the SyncML standard.

The SyncML standard was originally created to enable end users to seamlessly synchronize any type or format of network data with any mobile device. Since the creation of the data synchronization specification, the standards body has also developed the DM specification, which leverages the original specification. The two specifications are designed to be independent of the mobile device, network,

and applications. For this purpose SyncML utilizes Extensible Markup Language (XML) technology.

The first specification to evolve from the SyncML initiative was centered on synchronization of networked information to a mobile device. The goal of the common synchronization protocol was to solve the problem of multiple proprietary data synchronization protocols that limited the use and availability of data. The data synchronization specification enables any networked data to work with any mobile device. For example, an end user could access personal information, such as an address book, from a handheld or PDA and the data would be accessible and current.

Synchronization has different semantics when applied to firmware updates. For example, generic synchronization attempts to make an instance of data that looks like X into an instance of data that looks like X' (X-prime), and vice versa, without regard to the availability of various other versions of the same data, such as X" (X-prime-prime), etc. Firmware updates, in general, require the specification of what the target version should be.

In addition, firmware updates differ from synchronization in that the carrier-side or server-side does not have an entity that is a firmware—it only contains data that may be generated/translated into a firmware.

The second specification from the SyncML initiative revolves around the remote management of mobile devices, or device management (DM). Device management is the generic term used for technology that allows third parties to carry out the procedures of configuring mobile devices on behalf of the end user. SyncML defines device management as a technology that enables customization, personalization, and servicing of personal devices.

The DM specification proposes that future releases will facilitate software distribution, parameter configuration, and device capability verification. According to the SyncML White Paper, the planned device management scope includes device configuration (e.g. modifying or reading operating parameters), software maintenance (e.g. reading from a device its current operating parameters, reading a list installed or running software, hardware configurations), and diagnostics (e.g. listening for alerts sent from a device, invoking local diagnostics on a device).

The initial specification for the device management standard was released in February 2002. That release describes, from a high level, the protocol, component description, management frameworks, and security involved with the device management process. The DM specification has alluded to the fact that one of the future features would be the capability to download software. Details of the "software download" process were not included in the latest specification, but the download specification is projected to be completed in 2003. The standard does not specify formats of or methods to create update packages (encoded information of modifications to firmware/software necessary for upgrading to a new or modified version of the firmware/software) or how to apply the updates within the device. It mainly deals with management of the meta-data and the process around it. As with the aforementioned data synchronization protocol, the DM protocol will be open, and network and device agnostic.

SyncML has been promoted by the wireless industry as a solution for data transfers between mobile handsets and service providers. Although flexible, the current SyncML DM specifications do not yet provide support for firmware updates.

There is a fundamental problem in supporting firmware updates employing SyncML DM-based software. There is

no support in SyncML DM for firmware updates, as there are no commands or request structure that support setting of firmware version change indicators, power cycling, authentication/verification of software packages or firmware updates received, etc. In addition, the ability to request a software/firmware update in terms of source and target versions is essential but not currently supported in SyncML DM. Specifically, the semantics of retrieving a firmware update from a server on a carrier network is different from that of acquiring/retrieving generic data or content from a web site, and SyncML DM does not currently support these special semantics.

There is a fundamental asymmetry introduced into the client and server-sides of any SyncML-based solutions. Any "power cycling" command (needed to reinitialize a handset) that may be introduced into SyncML may apply only to the client-side, with SyncML core (or engines) on the server-sides configured to ignore such commands. In addition, any requests to update firmware may be restricted to the client-side.

Recent versions of SyncML DM do not have commands exclusively aimed at one side or the other, i.e. a command that is restricted to the client-side alone or the server-side alone. The SyncML DM management client may be a combination of a Download Agent with a security service and a resource manager that manages a tree of managed objects.

SyncML DM does not yet provide Update Agent (an agent for applying firmware updates in an electronic device) capabilities; it provides support for executing commands on the client-side. Such commands may include commands to Add, Copy, Delete, Exec, Get, Sequence, etc. that may be used to manipulate managed objects. Therefore, management objects may be added, copied, deleted, replaced, etc.

The SyncML management client may process XML, execute commands and transfer data. A SyncML DM client may comprise an XML parser that may employ SyncML DM elements and commands; an "engine" that may execute commands based on data retrieved from SyncML (XML) content; a tree or registry of management objects that may be targets of commands executed by the engine; and a data transport layer that may be based on one of Hyper Text Transfer Protocol (HTTP), Object Exchange (OBEX), Wireless Session Protocol (WSP), etc.

Further limitations and disadvantages of conventional and traditional approaches will become apparent to one of ordinary skill in the art through comparison of such systems with the present invention.

BRIEF SUMMARY OF THE INVENTION

Aspects of the present invention may be seen in a system employing Synchronization Markup Language (SyncML) device management specifications to facilitate firmware updates in an electronic device. The system comprises at least one electronic device having a memory, at least a portion of the memory comprising non-volatile memory containing firmware; a SyncML server communicatively coupled to the electronic device, the server comprising an enhanced SyncML DM server software; and a SyncML DM client resident in the electronic device, wherein the SyncML DM client is capable of interpreting enhancements to the SyncML DM specifications for updating the firmware.

The electronic device comprises communication software that supports at least one data transport protocol; a security module; and at least one software function that provides access to proprietary parameters in the electronic device.

The SyncML DM client comprises message processing software that facilitates processing and executing of SyncML messages, commands, alerts, and notifications; a SyncML manager; a download software that facilitates the downloading of at least one firmware update package from the SyncML server; and an update software that facilitates the updating of firmware using the at least one firmware update package.

The method for updating firmware in the electronic device in the system employing enhancements to SyncML DM specifications comprises sending a SyncML based notification to the electronic device; facilitating communication between the electronic device and the SyncML server; receiving the notification by a SyncML DM client resident in the electronic device; parsing the notification; and sending the notification for user review and subsequent user input. The method further comprises initiating a firmware update based on an input by the user; sending the firmware update to a download agent in the electronic device; communicating an appropriate SyncML message to initiate download of an update package from the SyncML server; facilitating and analyzing a response from the SyncML server; verifying validity and authentication of the update package, if an update package is received as part of the response; and dispatching commands in the response to appropriate modules.

These and other features and advantages of the present invention may be appreciated from a review of the following detailed description of the present invention, along with the accompanying figures in which like reference numerals refer to like parts throughout.

BRIEF DESCRIPTION OF SEVERAL VIEWS OF THE DRAWINGS

FIG. **1** illustrates a block diagram of an exemplary system with SyncML DM environment, in accordance with an embodiment of the present invention.

FIG. **2** illustrates an exemplary end-to-end architecture, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

FIG. **1** illustrates a block diagram of an exemplary system **105** with SyncML DM environment, in accordance with an embodiment of the present invention. The system **105** comprises a mobile handset **107** and a SyncML server **109**. The mobile handset **107** and the SyncML server **109** may be connected via a communications link **145**. The system **105** may employ enhancements to SyncML DM specifications to facilitate firmware updates on the mobile handset **107**. The mobile handset **107** may comprise a SyncML management client **111**, and a communication layer **113** that may provide one or more data transport protocols. In addition, the mobile handset **107** may comprise a security module **133**, and device wrappers **135** that may provide access to configuration parameters, hardware elements, memory elements and User Interface (UI) modules. In an embodiment of the present invention, the mobile handset **107** may also comprise a SIM card reader **131** and a SIM card **129**.

The SyncML server **109** may comprise a SyncML engine **137**. In an embodiment of the present invention, the SyncML server **109** may also comprise a provisioning interface **139**, which may provide an interface to one or more external service provisioning systems, a content database **141**, which may provide access to content such as firmware update

packages, and a notification manager **143**, which may provide notification support to facilitate notification of mobile handset **107**. The SyncML engine **137** may facilitate the creation and communication of SyncML based messages, notifications, update packages, etc. to the mobile handset **107**. The SyncML engine **137** may also support parsing and execution of SyncML requests including the enhancements to SyncML DM introduced by the present invention.

The SyncML management client **111** may comprise a SyncML engine **115**, a SyncML manager **123**, a download agent **125** and an update agent **127**.

The SyncML engine **115** may facilitate the processing and execution of SyncML messages, commands, alerts, notifications, etc. The SyncML engine **115** may comprise a parser **117** that may be used to parse incoming SyncML messages to retrieve commands and data, a command builder **119** that may be used to assemble SyncML messages, requests, notifications, alerts, etc., and a dispatcher **121** that may dispatch commands received in SyncML based messages, alerts, etc. for execution by the SyncML management client **111** or other modules or applications in the mobile handset **107**.

The SyncML manager **123** may provide support for managing the life-cycle of management objects, managing SyncML command execution, facilitating user interactions, etc. The download agent **125** may facilitate the download of update packages for software/firmware from the SyncML server **109** employing one or more data transport protocols supported by the communication layer **113**. The update agent **127** may facilitate the update of the firmware/software of the mobile handset by employing one or more downloaded update packages.

In an embodiment of the present invention, the SyncML server **109** may send a SyncML-based notification, to the mobile handset **107**, employing an external notification manager **143**. The notification may indicate the availability of a firmware update package. The communication layer **113** may facilitate the communication between the mobile handset **107** and the SyncML server **109**. The SyncML management client **111** may receive the notification message from the SyncML server **109**. The SyncML management client **111** may employ the parser **117** in the SyncML engine **115** to parse the received SyncML message. The dispatcher **121** may dispatch the notification for user review and subsequent user input, employing device wrappers **135**. The device wrappers **135** may be functions utilized to indirectly access proprietary information or code stored in hardware device (e.g., registers).

The dispatcher **121** may dispatch subsequent user-initiated update of firmware to the download agent **125**. The download agent **125** may communicate an appropriate SyncML message assembled by the command builder **119**, to initiate download of an update package from the SyncML server **109**. The download agent **125** may also facilitate the download of a response from the SyncML server **109**, the response subsequently being parsed by the parser **117** and analyzed by the SyncML manager **123** and the SyncML engine **115**. If an update package is received from the SyncML server **109** as part of the response, the SyncML manager **123** may ensure validity and authentication of the update package, by employing the security module **133** and the device wrappers **135**. The dispatcher **121** may dispatch the commands in the SyncML-based response to appropriate modules via device wrappers **135** or via the SyncML manager **123**. The commands may comprise commands such as, for example, a command for the verification of a received update package, a command for saving the update package in an appropriate management object, a command for initiating an update process by the update agent **127** and a command for subsequent notification of the result of update agent processing (success, failure, etc.).

In an embodiment of the present invention, the SyncML management client **111** may employ new commands for retrieving update packages for firmware updates and for updating the firmware based on appropriateness, security and authentication, employing fault tolerant update mechanism. The SyncML engine **137** in the SyncML server**109** may be capable of processing the new enhancement commands sent to it by the mobile handset **107**.

The update packages may be accompanied by a header that contains, among other entries, a cyclic redundancy check (CRC) value employed in the verification of the authenticity of the received update packages. The verification of received update packages may involve computing CRC values and comparing them to reference CRC values provided in a header that accompanies the update packages. Other forms of verification and authentication based on specific entries in the header are also contemplated.

In an embodiment of the present invention, a command to communicate the change of SIM card **129** detected by the mobile handset **107** may be supported by the mobile handset **107**. The SyncML engine **137** in the SyncML server **109** may be capable of processing such a command and acting upon it.

In an embodiment of the present invention, a client-side device such as, for example, a mobile handset **107**, may incorporate a management client **111** that may comprise an update agent **127**, a download agent **125** (simplified to ride on top of/employ one or more data transport protocols), a SyncML engine **115**, and a SyncML manager **123**. The SyncML engine **115** may comprise a SyncML dispatcher **121**, a SyncML parser **117**, and a SyncML command builder **119**.

In an embodiment of the present invention, a communication layer **113** may be utilized for the download agent **125** to interact with a management server such as, for example, the SyncML server **109**. The communication layer **113** may support such function as, for example, protocolInitialization, openConnection( ), closeConnection( ), sendData( ), and receiveData( ), on which the download agent may depend.

In an embodiment of the present invention a security layer may be utilized. The security layer may be considered to be external to the management client.

Aspects of the present invention retain differentiation in the creation of the update package and the process of applying the update fault tolerantly, as allowed by the SyncML standard. An embodiment of the present invention may utilize a newly defined, extensible, XML-based Agent-to-Server meta-data protocol that adapts more easily to the SyncML DM "software download" standard, when it is ratified. A package query protocol associated with an embodiment of the present invention may follow the same extensibility and meta-data management principles seen in current SyncML standards. As a result, various embodiments of the present invention may easily comply with the SyncML standard.

SyncML DM provides a management client that is responsible for providing access to management objects. The management objects may be arranged in a management tree. The management client may be capable of manipulating the management objects and the management tree. In addition, a management tree may be comprised of manage-

ment objects where each management object is a unit of code/data/application that may be accessed and/or manipulated.

The management client proposed by the SyncML standards is responsible for client-side support for SyncML based communications with management servers. Some of the duties associated with the management clients, include, for example, processing SyncML data, parsing, dispatching, and command building. A management client in an embodiment of the present invention may support dispatching and/or executing of SyncML DM commands such as, for example, alert, replace, status, add, copy, delete, exec, get, sequence, etc. The management client may also interact with a management server employing SyncML commands and structure using a communication toolkit. In addition, the management client proposed by the SyncML standards may support session initiation and termination.

The Alert command may be used to convey notifications, such as device management session requests, to the recipient. Such device management session requests may be client initiated or server initiated. In addition, wireless applications protocol(WAP) push-based notification may be a required feature in SyncML DM servers and clients, in an embodiment of the present invention.

SyncML DM clients may have to support security features such as hash message authentication code (HMAC), including basic authentication, challenge/responses, etc.

In an embodiment of the present invention, on the server-side, a device server may have its own SyncML engine to process user requests as well as to provide requested data to the mobile handsets in SyncML format. The device server may also be responsible for implementing the required security services, inserting security challenges in SyncML communications if necessary.

In an embodiment of the present invention, to be SyncML-compliant, an engine that executes SyncML DM commands may be provided. A tree of management objects may be created and maintained by the management client.

An embodiment of the present invention may comprise a SyncML management client. As a result, in addition to the download agent and the update agent, an embodiment of the present invention may employ XML parsers, encryption libraries, etc. in a mobile handset in order to assemble XML data/requests to be sent as SyncML DM commands/Alerts/Requests, parse any received XML data if data is returned as XML by the management server, and incorporate security mechanisms that may be used (Message Digest 5 (MD5), HMAC, etc.).

An embodiment of the present invention may enhance its set of device wrapper functions to be able to retrieve all the XML elements sent to the management server. The management client in such an embodiment may comprise its own download agent.

In accordance with the present invention, mobile handsets, which provide support for XML, may be multi-threaded. In a multi-threaded environment, it may not be appropriate to assume that an update process can immediately follow a download of an update package, because other applications may be running concurrently. As a result, an embodiment of the present invention may provide separate download and update processes.

An embodiment of the present invention may support reboot commands, reset commands, and power cycling by providing a location for storing downloaded update packages. An embodiment of the present invention may provide such a location by specifying a managed object for firmware updates.

An embodiment of the present invention may also provide an update package in a firmware updating service that comes up automatically during a power cycle/reboot. A SyncML DM command set enhancement that initiates a power cycling/reboot may be provided in an embodiment of the present invention. Another command enhancement may store a downloaded firmware update into an associated managed object. Yet another command enhancement may verify the authenticity of the firmware update package based on configurations and user profiles.

An embodiment of the present invention may utilize enhancement commands such as, for example, GetFirmwareUpdate, VerifyFirmwareUpdate, SaveFirmwareUpdate, ApplyFirmwareUpdate, ConfirmFirmwareUpdate, and SIMCardChange, explained below. An embodiment of the present invention may add these enhancement commands to the set of SyncML DM Protocol Command Elements as enhancements.

In an embodiment of the present invention, the SyncML DM protocol allows the enhancement commands to be executed on nodes of the management tree in the mobile device **107**. The SyncML device management may be the means by which the management tree is manipulated by the SyncML DM server. In an embodiment of the present invention, each node is addressed by a unique uniform resource identifier (URI). In SyncML DM protocol, the target and source of a command are identified by the target and source elements.

In an embodiment of the present invention, the SyncML management server may employ the exec command to invoke the enhancement commands associated with the firmware updates in the mobile handset. The exec command may launch a process that initiates the firmware update download in accordance with the parameters provided in the exec command. The implementation of the exec command may ensure that the entire firmware update file download is completed prior to starting any code rewrites on the device.

The command GetFirmwareUpdate may create a new management object that refers to an update package of the firmware. This command may specify an item element to be retrieved; the item element may be also specified with an associated target element. The firmware update retrieved may be saved as a management object.

The GetFirmwareUpdate command, in an embodiment of the present invention, may verify whether an update package already exists in the mobile handset before initiating a new download, in order to avoid an unnecessary download.

In an embodiment of the present invention, the GetFirmwareUpdate command may provide the functionality of the SyncML DM GET command, in addition to providing the support to verify whether the retrieval is necessary before conducting it.

In an embodiment of the present invention, the SyncML DM server may employ the GetFirmwareUpdate command to download the firmware update package from a given location such as a uniform resource locator (URL). The GetFirmwareUpdate command may be executed by the SyncML DM protocol using the exemplary XML code shown below:

```
<GetFirmwareUpdate>
    <CmdID>3</CmdID>
    <Item>
        <Target>
            <LocURI> 'Managed Object where update package stored is specified
            here' </LocURI>
        </Target>
        <Source>
            <LocURI>'URL of update package is specified here' </LocURI>
        </Source>
        <Data>DL_URL=AnyURL</Data>
        <Data>DL_Prompt=[0|1]</Data> <!-- 0 is no prompting, 1 is with prompting -->
    </Item>
</GetFirmwareUpdate>
```

The Target element of the GetFirmwareUpdate command specifies where in the management tree the downloaded update package gets inserted. The Data element DL_URL specifies the source URL for the download of the firmware update package. In an embodiment of the present invention, the command may be executed with a user prompt, in which case an appropriate value is employed in the Data element for DL_Prompt value. Other Data and/or Target elements may be also used.

In an embodiment of the present invention, other forms of download of the firmware update package may be utilized. For example, the following Download command, which may be a generic form of the GetFirmwareUpdate command, may be used with firmware as well as software and other forms of generic downloads:

```
<Download>
    <CmdID>3</CmdID>
    <Item>
        <Target>
            <LocURI> 'URI of a node for the storage of update package is
        specified here, or a node that represents the target of the update' </LocURI>
        <Target>
        <Source>
            <LocURI> 'URL of update package is specified here' </LocURI>
        </Source>
        <Data>DL_URL=AnyURL</Data>
        <Data>DL_Prompt=[0|1]</Data> <!-- 0 is no prompting, 1 is with prompting -->
    </Item>
</Download>
```

An embodiment of the present invention may support a VerifyFirmwareUpdate command to verify the authenticity and appropriateness of a downloaded firmware update. Appropriate results/errors may be communicated to the SyncML management client when appropriate. The VerifyFirmwareUpdate command may be executed by the SyncML DM protocol using the exemplary XML code shown below:

```
    <VerifyFirmwareUpdate>
        <CmdID>3</CmdID>
        <Item>
```

-continued

```
            <Target>
                <LocURI>./FWUpdate/x/PkgStatus</LocURI>
            </Target>
            <Source>
                <LocURI>./FWUpdate/x/PkgURL</LocURI>
            </Source>
        </Item>
    </VerifyFirmwareUpdate>
```

The VerifyFirmwareUpdate command may verify the specified update package. The update package may be specified using the Source element, and the result of such verification may be populated into a Target URI. Other Target and/or Source elements may also be used.

An embodiment of the present invention may support a SaveFirmwareUpdate command to save the firmware update in an appropriate management object and/or FLASH segments in memory. The SaveFirmwareUpdate command, in an embodiment of the present invention, may provide support for the functionality of the SyncML DM Replace or Add commands. The SaveFirmwareUpdate command may be used to ensure that an update agent sets an "update" bit in FLASH for a subsequent update process. The SaveFirmwareUpdate command may be executed by the SyncML DM protocol using the exemplary XML code shown below:

```
<SaveFirmwareUpdate>
    <CmdID>3</CmdID>
    <Item>
        <Source>
            <LocURI>./FWUpdate/x/PkgData</LocURI>
        </Source>
        <Data>Address=0x00089F</Data> <!—Address specified as hex value-->
    </Item>
</SaveFirmwareUpdate>
```

The SaveFirmwareUpdate command facilitates saving of the update package in specific address locations that may be implementation-dependent. In the above example, binary data from the management tree located in the mobile handset at the location ./FWUpdate/x/PkgData is saved at the address specified in the Data element for 'Address' parameter. Other Source and/or Data elements may also be used.

An embodiment of the present invention may support an ApplyFirmwareUpdate command to take a target firmware update management object and launch an update process. The update process may involve power cycling the mobile handset. The result of taking a target firmware update management object and launching an update process may be communicated only after the power cycling step has been executed. The ApplyFirmwareUpdate command may be executed by the SyncML DM protocol using the exemplary XML code shown below:

```
<ApplyFirmwareUpdate>
    <CmdID>3</CmdID>
    <Item>
        <Source>
            <LocURI>./FwUpdate/x/PkgURL</LocURI>
        </Source>
        <Data>argument1</Data>
        <Data>argument2</Data>
        <Data>argument3</Data>
    </Item>
</ApplyFirmwareUpdate>
```

The ApplyFirmwareUpdate command facilitates the invocation of an update agent. The ApplyFirmwareUpdate command specifies the Source element as a URI in the management tree, and specifies optional prompts. Other Source and/or Data elements may also be used.

In an embodiment of the present invention, other Update commands may be utilized for updating firmware with a firmware update package. For example, the following Update command, which may be a generic form of the ApplyFirmwareUpdate command, may be used:

```
<Update>
    <CmdID>3</CmdID>
    <Item>
        <Target>
            <LocURI>./.../Application/ ApplicationXXX</LocURI>
        </Target>
        <Source>
            <LocURI>./FwUpdate/x/PkgURL</LocURI>
        </Source>
        <Data>argument1</Data>
        <Data>argument2</Data>
        <Data>argument3</Data>
    </Item>
</Update>
```

In the Update command, Target specifies the node in the management tree that represents the Application or component to be updated. The Source specifies the URL from which the update package(s) are to be downloaded or retrieved. A node that provides an update package that may have been previously downloaded, or may be part of the management tree, may also be employed.

An embodiment of the present invention may support a ConfirmFirmwareUpdate command to confirm the status of the previously executed firmware update event. The ability to confirm successful completion of a firmware update may be used to initiate billing activities on the carrier-side. The ConfirmFirmwareUpdate command, in an embodiment of the present invention, may support the functionality of a SyncML DM generic update confirmation command/notification, which may be utilized to indicate a successful completion of update events, specifically to initiate possible billing activities.

When a SIM card changes on the mobile handset side, the nature of the service may change on the carrier-side. An embodiment of the present invention may support a SIMCardChange command to facilitate the notification of service-level changes with possible changes to billing events.

FIG. **2** illustrates an exemplary end-to-end architecture, in accordance with an embodiment of the present invention. The solution depicted in FIG. **2** is compliant with SyncML specifications.

A mobile handset **219** may comprise firmware. The mobile handset **219** may be communicatively coupled with a SyncML DM server **209** via a communication link **213**. The communication link between the SyncML DM server **209** and the mobile handset **219** may utilize a SyncML protocol **213**. The SyncML DM server **209** may receive a firmware update package **207** from the generator **205** that generated the update package **207** utilizing an old (original) version of the firmware **201** and a new (target) version of the firmware **203**.

In an embodiment of the present invention, the SyncML DM server **209** may be used to store the update packages **207**, to manage delivery of the update packages **207**, and to maintain system security. The enhancement commands described hereinabove enable an enhanced SyncML DM server **209** to provide the firmware updating service, which is not supported by the commands available using a server compliant only with the existing SyncML DM standards.

In an embodiment of the present invention, the mobile handset **219** may comprise a SyncML DM download agent **215**, and an update agent **217**. The SyncML DM download agent **215** may be responsible for initializing, transferring, and checking the update package **207** sent by the SyncML DM server **209**. The update agent **217** may then apply the updates to the firmware in a fault tolerant manner in accordance with an embodiment of the present invention.

While the present invention has been described with reference to certain embodiments, it will be understood by

those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the present invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the present invention without departing from its scope. Therefore, it is intended that the present invention not be limited to the particular embodiment disclosed, but that the present invention will include all embodiments falling within the scope of the appended claims.

What is claimed is:

1. A system employing Synchronization Markup Language (SyncML) device management specifications, recorded in memory and capable of being processed by an electronic device, to facilitate firmware updates in the electronic device, the system comprising:

at least one electronic device having a memory, at least a portion of the memory comprising non-volatile memory containing firmware;

a SyncML server communicatively coupled to the electronic device, the server comprising an enhanced SyncML DM server software; and

a SyncML DM client resident in the electronic device, wherein the SyncML DM client is capable of interpreting enhancements to the SyncML DM specifications for updating the firmware.

2. The system according to claim 1 wherein the electronic device comprises:

communication software that supports at least one data transport protocol;

a security module; and

at least one software function that provides access to proprietary parameters in the electronic device.

3. The system according to claim 2 wherein the electronic device further comprises a security device.

4. The system according to claim 2 wherein the electronic device further comprises a security device reader.

5. The system according to claim 2 wherein the SyncML DM client comprises:

message processing software that facilitates processing and executing of SyncML messages, commands, alerts, and notifications;

a SyncML manager;

a download software that facilitates the downloading of at least one firmware update package from the SyncML server; and

an update software that facilitates the updating of firmware using the at least one firmware update package.

6. The system according to claim 5 wherein the message processing software comprises:

a first software that parses SyncML messages to retrieve data;

a second software that assembles SyncML messages; and

a third software that sends the data retrieved from the SyncML messages for execution.

7. The system according to claim 1 wherein the SyncML server comprises a SyncML engine.

8. The system according to claim 7 wherein the SyncML server further comprises an interface to at least one external service provisioning system.

9. The system according to claim 7 wherein the SyncML server further comprises a manager that facilitates notification of the electronic device.

10. The system according to claim 7 wherein the SyncML engine facilitates the creation and communication of SyncML messages and notifications to the electronic device.

11. The system according to claim 7 wherein the SyncML engine facilitates the creation and communication of update packages to the electronic device.

12. The system according to claim 7 wherein the SyncML engine supports parsing and executing at least one enhancement of SyncML requests such as the enhancements to SyncML device management specifications.

13. The system according to claim 7 wherein the SyncML server further comprises a database that provides access to copies of the firmware in the electronic device.

14. The system according to claim 13 wherein the content is firmware update packages.

15. A method for updating firmware in an electronic device in a system employing enhancements to SyncML DM specifications, recorded in memory and capable of being processed by an electronic device, the system comprising the electronic device and a SyncML server, the method comprising:

receiving, by a SyncML DM client resident in the electronic device, a SyncML based notification from the SyncML server;

parsing the notification; and

sending the notification for user review and subsequent user input.

16. The method according to claim 15 wherein the notification indicates availability of a firmware update package.

17. The method according to claim 15 wherein the method further comprises:

initiating a firmware update based on an input by the user;

sending the firmware update to a download agent in the electronic device;

communicating an appropriate SyncML message to initiate download of an update package from the SyncML server; and

facilitating and analyzing a response from the SyncML server.

18. The method according to claim 17 further comprising:

verifying validity and authentication of the update package, if an update package is received as part of the response; and

dispatching commands in the response to appropriate modules.

19. The method according to claim 18 wherein the commands comprise a command for verification of the received update package.

20. The method according to claim 18 wherein the commands comprise a command for saving the update package in an appropriate management object.

21. The method according to claim 18 wherein the commands comprise a command for retrieving update packages.

22. The method according to claim 18 wherein the commands comprise a command for updating the firmware based on appropriateness, security, and authentication.

23. The method according to claim 18 wherein the commands comprise a command for initiating an update process by the update agent.

24. The method according to claim 23 wherein the commands comprise a command for subsequent notification of the result of the update agent processing.

25. The method according to claim 17 wherein the SyncML message is assembled in the electronic device.

26. A mobile electronic device comprising: machine-readable storage containing SyncML DM interpreter code executable by the mobile electronic device; and

Wherein the SyncML DM interpreter code supports updates and downloads of software and firmware in the mobile electronic device.

* * * * *

SyncML Meta-Information DTD, version 1.0

Abstract

This document defines a XML Document Type Definition (DTD) as defined in [5]. The DTD represents standard meta-information used in the SyncML Representation Protocol as defined in [3]. The meta-information is included in a SyncML XML document through the declaration of the SyncML Meta-information DTD name space on any element types from this DTD. See [3] for more details.

SyncML Initiative

The following companies are Sponsors of the SyncML initiative:

Ericsson
IBM
Lotus
Matsushita Communications Industrial Co.
Motorola
Nokia
Palm, Inc.
Psion
Starfish Software

Revision History

Revision	Date	Comments
v0.9	2000-05-31	Version 0.9.
v1.0a	2000-08-24	Completed description of each element. Updated format for headers. Included correct Sponsor name. Added WBXML tokens.
V1.0b	2000-11-13	Added MaxMsgSize element. Updated URL in reference section.
V1.0	2000-12-07	Candidate version for the final release.

Copyright Notice

Table of Contents

1 Introduction

This Document Type Definition (DTD) defines a set of mark-up that is used by the SyncML DTD to identify meta-information associated with a SyncML command or data item or collection.

2 Formatting Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interoperated as described in [2].

Any reference to components of the Device Information DTD or XML snippets is specified in this `typeface`.

3 Terminology

Meta-Information

Parameter or attributes about the representation, state or type or content of an object or property.

4 XML Usage

The SyncML **Meta-Information** is represented in a mark-up language defined by [5]. The meta-information is identifiable as an XML name space. The SyncML Meta-Information DTD (Document Type Definition) defines the XML document type used to represent meta-information used in the [3] representation protocol. The SyncML Meta-Information DTD can be found in Section 5.

The SyncML Meta-Information XML documents are specified using well-formed XML. However, they need not be valid XML. That is, they do not need to specify the XML prolog. They only need to specify properly identified name space element types from the SyncML Meta-Information DTD. This restriction allows for the SyncML Meta-Information to be specified with greater terseness than would be possible if a well-formed, valid XML document was required.

This DTD makes heavy use of XML name spaces. Name spaces MUST be declared on the first element type that uses an element type from the name space. Element types from the SyncML Meta-Information DTD can be used in other XML documents, including a SyncML message.

Names in XML are case sensitive. By convention in the SyncML Meta-Information DTD, the element type and attribute list names are specified with a "Hungarian" like notation of the first character in each word of the name in upper case text and remainder of the characters in each word of the names specified in lower case text. For example, `MetInf` for the Sync meta-information root element type or `Type` for the content type tag.

The element types in the SyncML Meta-Information DTD are defined within a namespace associated with the URI `http://www.syncml.org/docs/syncml_metinf_v10_20001207.dtd` or the URN `syncml:metinf`.

SyncML also makes use of XML standard attributes, such as `xml:lang`. Any XML standard attribute can be used in a XML document conforming to this DTD.

XML can be viewed as more verbose than alternative binary representations. This is often cited as a reason why it may not be appropriate for low bandwidth network protocols. In most cases, this DTD uses shortened element type and attribute names. This provides a minor reduction in verbosity. Additionally, the SyncML Meta-Information can be encoded in a tokenized, binary format defined by [4]. The use of [4] format is external to specification of the DTD and should be transparent to any XML application. The combination of the use of shortened element type names and an alternative binary format makes this DTD competitive, from a compressed format perspective, with alternative, but private, binary representations.

One of the main advantages of XML is that it is a widely accepted International recommendation for text document mark-up. It provides for both human readability and machine processability. In addition, XML allows the originator to capture the structure of a document, not just it's content. This is extremely useful for applications such as data synchronization, where not just content, but structure semantics is often exchanged.

5 Element Type Descriptions

5.1 Anchor

Usage: Specifies the synchronization state information (i.e., sync anchor) for the current synchronization session.

Parent Elements: `MetInf`

Restrictions: The optional `Last` element type specifies the synchronization anchor for the previous synchronization session. The required `Next` element type specifies the synchronization anchor for the current synchronization session.

Content Model:

```
(Last?, Next)
```

Attributes: None.

Example:

```
<Anchor xmlns='syncml:metinf'>
  <Last>20000824T133000Z</Last><Next>20000824T221300Z</Next>
</Anchor>
```

5.2 Format

Usage: Specifies the encoding format of the content information in the `Data` element.

Parent Elements: `MetInf`

Restrictions: The value of this element SHOULD BE one of `b64`, `chr`, `int`, `null` or `xml`. If the element type is missing, the default value is `chr`. If the value is `chr`, then the format of the content information is clear-text in the character set specified on either the transport protocol, the MIME content type header or the XML prolog. If the value is `int`, then the format of the content information is numeric text representing an unsigned integer between zero and 2**32-1. If the value is `null`, then there is no content information. This value is used by some synchronization data models to delete the content, but not the presence of the property. If the value is `b64`, then the format of the content information is binary data that has been character encoded using the Base64 transfer encoding defined by [RFC2045]. If the value is `xml`, then the format of the content information is XML structured mark-up data.

The target object is the one in which the meta-information appears.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following example illustrates how the element type is used within the SyncML DTD to specify format meta-information for data in the Item element type.

```
<Item>
  <Meta>
    <Format xmlns='syncml:metinf'>int</Format>
  </Meta>
  <Data>1024</Data>
</Item>
```

5.3 Last

Usage: Specifies the synchronization state information (i.e., sync anchor) for the last successful synchronization session.

Parent Elements: `MetInf`

Restrictions: The value MUST specify either an UTC based ISO 8601 [1] date/time stamp or a monotonically increasing numeric integer string. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by [ISO8601].

Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<Anchor xmlns='syncml:metinf'>
  <Last xmlns='syncml:metinf'>20000824T133000Z</Last>
  <Next xmlns='syncml:metinf'>20000824T221300Z</Next>
</Anchor>
```

5.4 Mark

Usage: Specifies a meta-information "mark" on the data object.

Parent Elements: `MetInf`

Restrictions: The content information for this element type SHOULD BE one of `draft`, `final`, `delete`, `undelete`, `read`, `unread`.

When this meta-information is specified repetitively in a hierarchically of element types (e.g., in a SyncML collection, as well as the items in the collection), then the meta-information specified in the lowest level element type takes precedence.

This element type is used to set the meta-information characteristics of a data object, such as the draft/final, delete/undelete, read/unread marks on a folder item or mail item.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following example illustrates how the element type is used within the SyncML DTD to specify meta-information about a data object specified by the Item element type.

```
<Update>
  <CmdID>10</CmdID>
  <Item>
    <Source>
      <LocName>jsmith</LocName>
      <LocURI>host1.com-19991208T234504-001</LocURI>
    </Source>
    <Meta>
      <Mark xmlns='syncml:metinf'>unread</Mark>
    <Meta>
  </Item>
</Update>
```

5.5 MaxMsgSize

Usage: Specifies the maximum byte size of any response message to a given SyncML request.

Parent Elements: Root element type.

Restrictions: The element type appears in the `Meta` element in the `SyncHdr` of a SyncML request to specify the maximum size of any subsequent response messages. The element type is usually specified by a SyncML client, but can also be specified by a SyncML server.

This element type value is applicable for the remainder of the synchronization session, unless it is specified again.

The element type value is the text string representation of the maximum, decimal byte size of any response message.

In order to use the elements from the MetInf name space, the root element does not need to be specified.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: Normally, the root element type does not appear in a SyncML `Meta` element type.

```
<MaxMsgSize xmlns='syncml:metinf'>1023</MaxMsgSize>
```

5.6 MetInf

Usage: Specifies the root element for the SyncML meta-information document.

Parent Elements: Root element type.

Restrictions: In order to use the elements from the MetInf name space, the root element does not need to be specified. The element type can appear in the `Meta` element of a SyncML document to allow for declaring a default name space.

Content Model:

```
(Format?, Type?, Mark?, Size?, Anchor?, Version, NextNonce?)
```

Attributes: None.

Example: Normally, the root element type does not appear in a SyncML `Meta` element type.

```
<mi:MetInf xmlns:mi='syncml:metinf'>
  <mi:Type>text/calendar</mi:Type>
  <mi:Format>chr</mi:Type>
  <mi:Size>877566</mi:Size>
  <mi:Version>20000714T082300Z</mi:Version>
</mi:MetInf>
```

5.7 Next

Usage: Specifies the synchronization state information (i.e., sync anchor) for the current synchronization session.

Parent Elements: `MetInf`

Restrictions: The value MUST specify either an UTC based ISO 8601 [1] date/time stamp or a monotonically increasing numeric integer string. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by [ISO8601].

Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<Anchor xmlns='syncml:metinf'>
  <Last xmlns='syncml:metinf'>20000824T133000Z</Last>
  <Next xmlns='syncml:metinf'>20000824T221300Z</Next>
</Anchor>
```

5.8 NextNonce

Usage: Specifies the nonce string to be used in any subsequent communication.

Parent Elements: `MetInf`

Restrictions: The nonce string MUST be further re-formatted using the Base64 algorithm.

This element type is used to specify the next nonce string that is to be used in any subsequent SyncML message. For example, a SyncML server specifies this element type to tell the SyncML client to change it's nonce to a new value.

Nonce strings are used in the SyncML "MD5 Digest" scheme of authentication credentials.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example:

```
<Meta>
  <NextNounce
xmlns='syncml:metinf'>QWxhZGRpbjpvcGVuIHNlc2FtZQ==</NextNounce>
<Meta>
```

5.9 Size

Usage: Specifies the byte size of a data object.

Parent Elements: `MetInf`

Restrictions: The byte size is specified as the numeric text equivalent of the byte count of the data object.

Content Model:

```
(#PCDATA)
```

Attributes: None

Example: The following example illustrates how the element type is used within the SyncML DTD to specify meta-information about the byte size of the Item element type.

```
<Item>
  <Target><LocURI>4</LocURI>
  <Meta>
  <Size xmlns='syncml:metinf'>10</Size>
  </Meta>
  <Data>John Smith</Data>
</Item>
```

5.10 Type

Usage: Specifies the media type of the content information in the `Data` element.

Parent Elements: `MetInf`

Restrictions: If this element is missing, then the default content-type is `text/plain`. The content information for this element type SHOULD BE a registered MIME content-type. Alternatively, a URN can be used to specify the media type.

Content Model:

```
(#PCDATA)
```

Attributes: None

Example: The following example illustrates how the element type is used within a SyncML message to specify meta-information about the media type of the content information in the Item element type.

```
<Item>
  <Target><LocURI>3</LocURI></Target>
  <Meta>
    <Type xmlns='syncml:metinf'>text/directory;profile=vCard</Type>
  </Meta>
  <Data>BEGIN:VCARD
VERSION:3.0
FN:Jim Smith
N:Smith;Jim
TEL;TYPE=WORK,VOICE,FAX:+1-919-555-1234
EMAIL;TYPE=INTERNET,WORK:Jim_Smith@mail.host.com
END:VCARD
  </Data>
</Item>
```

5.11 Version

Usage: Specifies the revision identifier of a data object.

Parent Elements: `MetInf`

Restrictions: The value MUST specify either an UTC based ISO 8601 [1] date/time stamp or a monotonically increasing numeric integer string. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by [ISO8601]. Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object.

If not present, then the object doesn't currently have a revision version identifier. When the element type is missing, this SHOULD NOT be interpreted as meaning the original version of the data object.

Content Model:

```
(#PCDATA)
```

Attributes: None.

Example: The following example illustrates how the element type is used within the SyncML DTD to specify meta-information about the synchronization version of the Item element type.

```
<Item>
  <Target><LocURI>4</LocURI>
  <Meta>
  <Version xmlns='syncml:metinf'>20000301T133000Z</Version>
  </Meta>
  <Data>John Smith</Data>
</Item>
```

6 DTD Definition

```
<!--Copyright Notice

Copyright (c) 2000 Ericsson, IBM, Lotus, Matsushita Communications
Industrial Co., Motorola, Nokia, Palm, Inc., Psion, Starfish Software. All
Rights Reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any,
must include the following acknowledgement: This product includes software
developed by The SyncML Initiative.

Alternatively, this acknowledgment may appear in the software itself, if
and wherever such third-party acknowledgment normally appears.
```

```
4. The name of the authors, jointly or severally, must not be used to
endorse or promote products derived from this software without specific
prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS ``AS IS'' AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR NON-
INFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

-->

<!-- This DTD defines a sequence of meta-information that is used within
the SyncML DTD.

This DTD is to be identified by the URI string "syncml:metainf".

Single element types from this name space can be referenced as follows:

    <element xmlns='syncml:metinf'>blah, blah</element>

-->

<!-- Root or Document Element and -->

<!ELEMENT MetInf (Format?, Type?, Mark?, Size?, Anchor?, Version?,
NextNonce?, MaxMsgSize?)>

<!-- Format or encoding type -->

<!ELEMENT Format (#PCDATA)>

<!-- Element specific type specification -->

<!ELEMENT Type (#PCDATA)>

<!-- Mark -->

<!ELEMENT Mark (#PCDATA)>

<!-- Byte count -->

<!ELEMENT Size (#PCDATA)>

<!-- Data versioning info -->

<!ELEMENT Anchor (Last?, Next)>

<!ELEMENT Last (#PCDATA)>

<!ELEMENT Next (#PCDATA)>
```

```
<!ELEMENT Version (#PCDATA)>

<!ELEMENT NextNonce (#PCDATA)>

<!ELEMENT MaxMsgSize (#PCDATA)>

<!-- End of DTD →
```

7 WBXML Definitio n

The following tables define the token assignments for the mapping of the Meta-Information DTD element types into WBXML as defined by [4].

7.1 Code Space and C ode Page Definitions

The element types from the Meta-Information DTD (i.e., name space) are only intended to be used within a SyncML document. Hence, the elements from the Meta-Information DTD are always maped into the single SyncML WBXML code space.

DTD Name	WBXML PUBLICID Token (Hex Value)	Formal Public Identifier
SyncML	FD1	-//SYNCML//DTD SyncML 1.0//EN

The SyncML DTD is assigned the WBXML document public identifier (i.e., the "publicid" WBXML BNF production) associated with the `FD1` token.

The element types from the Meta-Information DTD utilize the code page x02 (two) within the SyncML Code Space.

DTD Name	WBXML Code Page Token (Hex Value)	Formal Public Identifier
SyncML	00	-//SYNCML//DTD SyncML 1.0//EN
MetInf	02	-//SYNCML//DTD MetInf 1.0//EN

7.2 Token Definitions

The following WBXML token codes represent element types (i.e., tags) form code page x02 (two), Meta-Information DTD.

Element Type Name	WBXML Tag Token (Hex Value)
Anchor	05
Format	06
Last	07
Mark	08
MetInf	09
Next	0A
NextNonce	0B
Size	0C
Type	0D
Version	0E
MaxMsgSize	0F

The WBXML token codes from code page x01 (one) represent the DevInf DTD. These token definitions are defined in the DevInf DTD specification.

The WBXML token codes from code page x02 (two) represent the MetInf DTD. These token definitions are defined in the MetInf DTD specification.

8 Static Conformance Requirements

Static conformance requirements (SCR) specify the features that are optional, mandatory and recommended within implementations conforming to this specification.

Simple tables are used to specify this information

In these tables, optional features are specified by a "MAY", mandatory features are specified by a "MUST" and recommended features are specified by a "SHOULD".

The following specifies the static conformance requirements for the SyncML Meta-Information element types for devices conforming to this specification.

Element Type	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
Anchor	MUST	MUST	MUST	MUST
Format	MUST	MUST	MUST	MUST
Last	MUST	MUST	MUST	MUST
Mark	MAY	MAY	MAY	MAY
MaxMsgSize	MAY	MUST	MAY	MUST
MetInf	MUST	MUST	MUST	MUST
Next	MUST	MUST	MUST	MUST
NextNonce	MUST	MUST	MUST	MUST
Size	MAY	MAY	MAY	MAY
Type	MUST	MUST	MUST	MUST
Version	MUST	MUST	MAY	MAY

9 References

[1] Data elements and interchange formats - Information interchange - Representation of dates and times, ISO.

[2] Key words for use in RFCs to Indicate Requirement Levels, IETF.

[3] SyncML Representation Protool DTD, SyncML.

[4] WAP Binary XML Content Format Specification, WAP Forum.

[5] Extensible Mark-up Language (XML) 1.0, W3C.

US007188143B2

(12) **United States Patent**
Szeto

(10) **Patent No.: US 7,188,143 B2**
(45) **Date of Patent: *Mar. 6, 2007**

(54) **MESSENGER-CONTROLLED APPLICATIONS IN AN INSTANT MESSAGING ENVIRONMENT**

(75) Inventor: **Christopher Tzann-en Szeto**, Santa Clara, CA (US)

(73) Assignee: **Yahoo! Inc.**, Sunnyvale, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 230 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/613,985**

(22) Filed: **Jul. 2, 2003**

(65) **Prior Publication Data**

US 2004/0215731 A1 Oct. 28, 2004

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/930,878, filed on Aug. 15, 2001, now Pat. No. 7,133,900.

(60) Provisional application No. 60/331,331, filed on Jul. 6, 2001.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/206**; 709/202; 709/204; 709/227; 715/751; 715/758; 707/104.1

(58) **Field of Classification Search** 709/227–228, 709/204, 217–219, 238, 200, 201–207, 223–224, 709/250, 246–247; 719/328, 329; 370/259, 370/352, 353, 389, 400; 715/751–753, 758; 707/3, 10, 104.1; 455/518–519
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,220,657 A 6/1993 Bly et al.
5,880,731 A 3/1999 Liles et al.
6,237,025 B1 5/2001 Ludwig et al.
6,301,609 B1 10/2001 Aravamudan et al.
6,430,602 B1 8/2002 Kay et al.
6,487,583 B1 * 11/2002 Harvey et al. 709/204
6,493,724 B1 12/2002 Cusack et al.
6,539,421 B1 3/2003 Appleman et al.
6,564,246 B1 5/2003 Varma et al.
6,564,249 B2 5/2003 Shiigi
6,611,814 B1 8/2003 Lee et al.
6,651,053 B1 11/2003 Rothschild

(Continued)

OTHER PUBLICATIONS

International Searching Authority, "Notification of transmittal of the international search report and the written opinion of the international searching authority, or the declaration," International Application No. PCT/US2004/038354, dated Apr. 19, 2005, 13 pages.

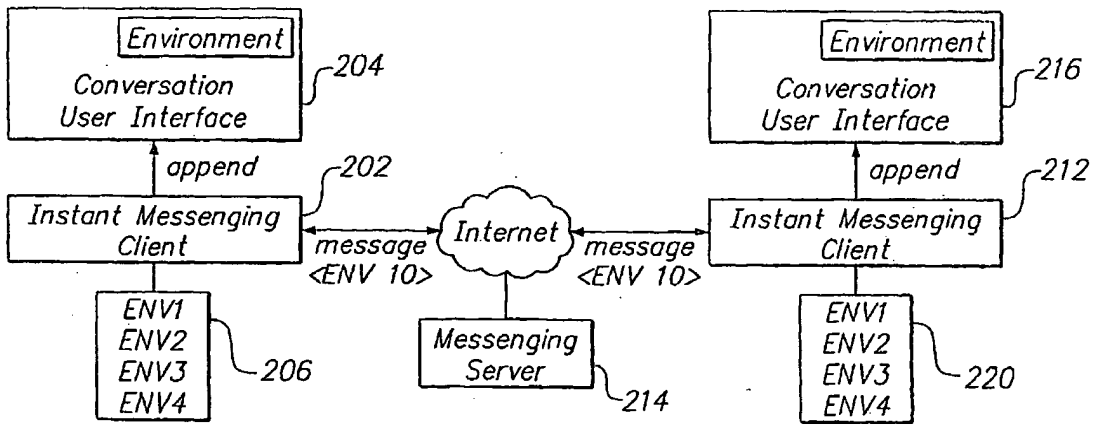
(Continued)

Primary Examiner—Jeffrey Pwu
(74) *Attorney, Agent, or Firm*—Hickman Palermo Truong & Becker LLP

(57) **ABSTRACT**

Techniques for controlling an application in an instant messaging environment are described. The instant messaging environment retrieves and executes an instant messaging application. An identifier is assigned to the instant messaging application, which is retrieved and executed in the instant messaging environment between two or more instant messaging clients. The instant messaging environment identifies a selected instant messaging application and generates a control message. The control message includes the identifier which is used to retrieve and execute the instant messaging application. The instant messaging environment also determines whether a supporting application is required, based on the instant messaging application identifier, to execute the instant messaging application.

14 Claims, 18 Drawing Sheets



U.S. PATENT DOCUMENTS

6,677,976	B2 *	1/2004	Parker et al.	348/14.08
6,747,970	B1	6/2004	Lamb et al.	
6,760,580	B2	7/2004	Robinson et al.	
6,781,608	B1 *	8/2004	Crawford	715/758
6,807,565	B1 *	10/2004	Dodrill et al.	709/206
6,816,884	B1	11/2004	Summers	
6,907,447	B1	6/2005	Cooperman et al.	
6,980,983	B2	12/2005	Banerjee et al.	
6,983,370	B2	1/2006	Eaton et al.	
7,028,262	B2	4/2006	Estrada et al.	
2003/0041108	A1	2/2003	Henrick et al.	
2003/0101235	A1	5/2003	Zhang	
2003/0208545	A1	11/2003	Eaton et al.	
2004/0117443	A1	6/2004	Barsness	
2004/0215731	A1	10/2004	Tzann-en Szeto	
2005/0086211	A1	4/2005	Mayer	

OTHER PUBLICATIONS

Current Claims, PCT/US2004/038354, 3 pages.

"Private-Label Instant Messaging Clients", Odigo Inc.. Retrieved from the Internet: http://corp.odigo.com/products/clients downloaded Oct. 9, 2001.

"Quicktime", Apple Computer, Inc. (2001) Retrieved from the Internet. http://www.apple.com/quicktime/ downloaded Oct. 10, 2001.

International Searching Authority, "Notification of the Transmittal pf the International Search Report and the Written Opinion of the International Searching Authority, or the Declaration," PCT/US04/21209, dated Aug. 09, 2005, 9 pages.

Current Claims, PCT/US04/21209, 4 pages.

http://corp.odigo.com/products/clients Web Page.

http://www.apple.com/quicktime/ Web Page.

"Instant Messaging: Convenient Chatting and Corporate Battling" retrieved on May 22, 2006 from the Internet<URL: http://www.glencoe.com/norton/n-instructor-/updates/1999/9799-4.html > 5 pages.

* cited by examiner

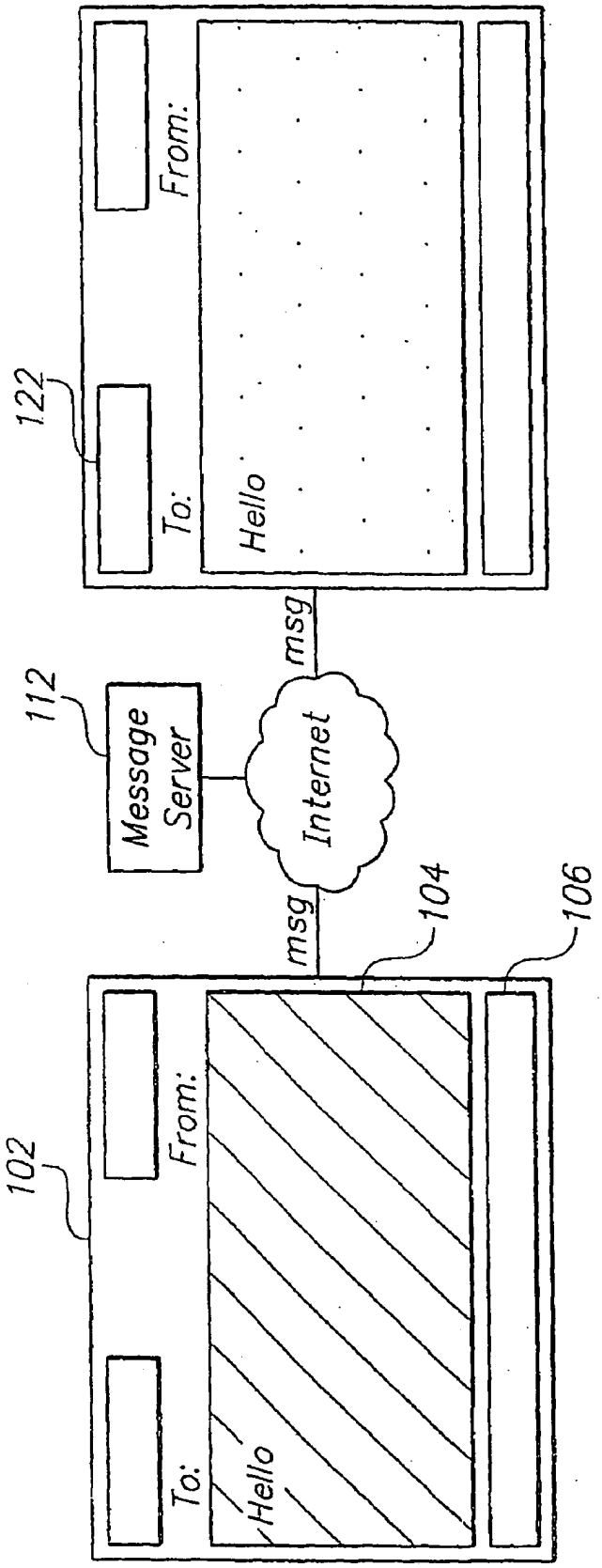


FIG. 1
(Prior Art)

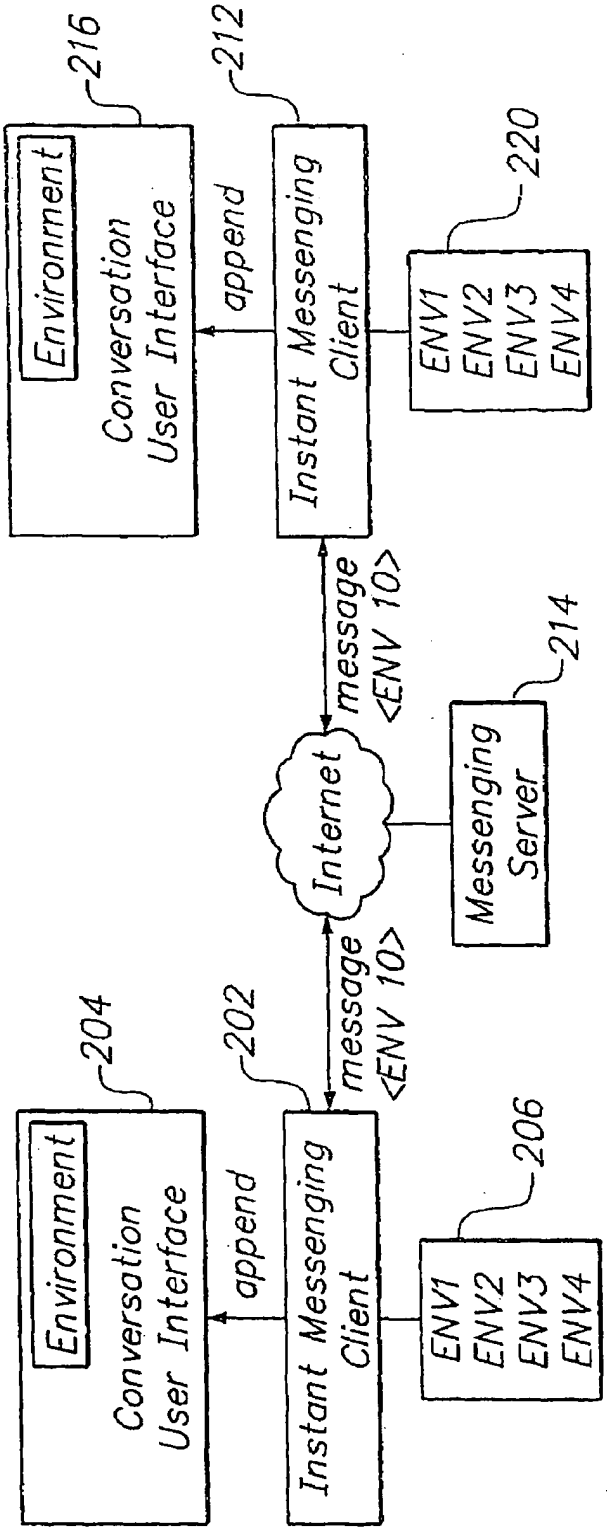

216
Environment
Conversation User Interface
212
append
Instant Messenging Client
220
ENV1
ENV2
ENV3
ENV4
message <ENV 10>
214
Internet
Messenging Server
message <ENV 10>
204
Environment
Conversation User Interface
202
append
Instant Messenging Client
206
ENV1
ENV2
ENV3
ENV4

FIG. 2

FIG. 3

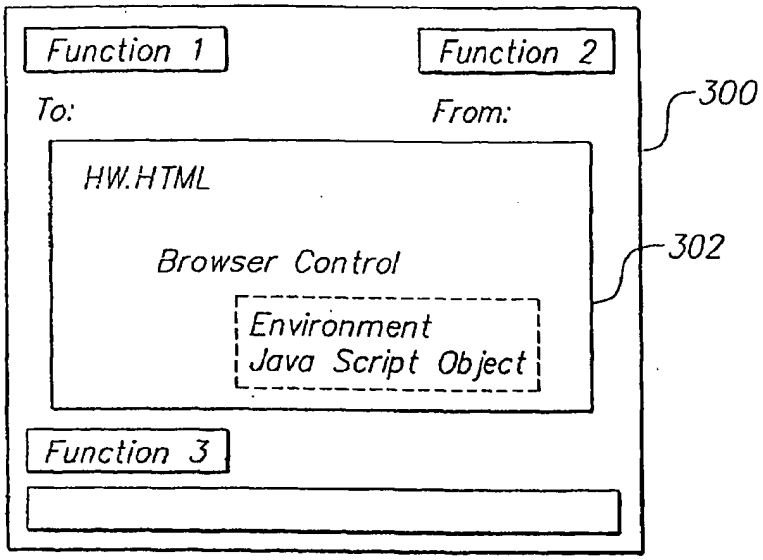

Function 1
Function 2
To:
From:
300
HW.HTML
Browser Control
302
Environment
Java Script Object
Function 3

FIG. 4

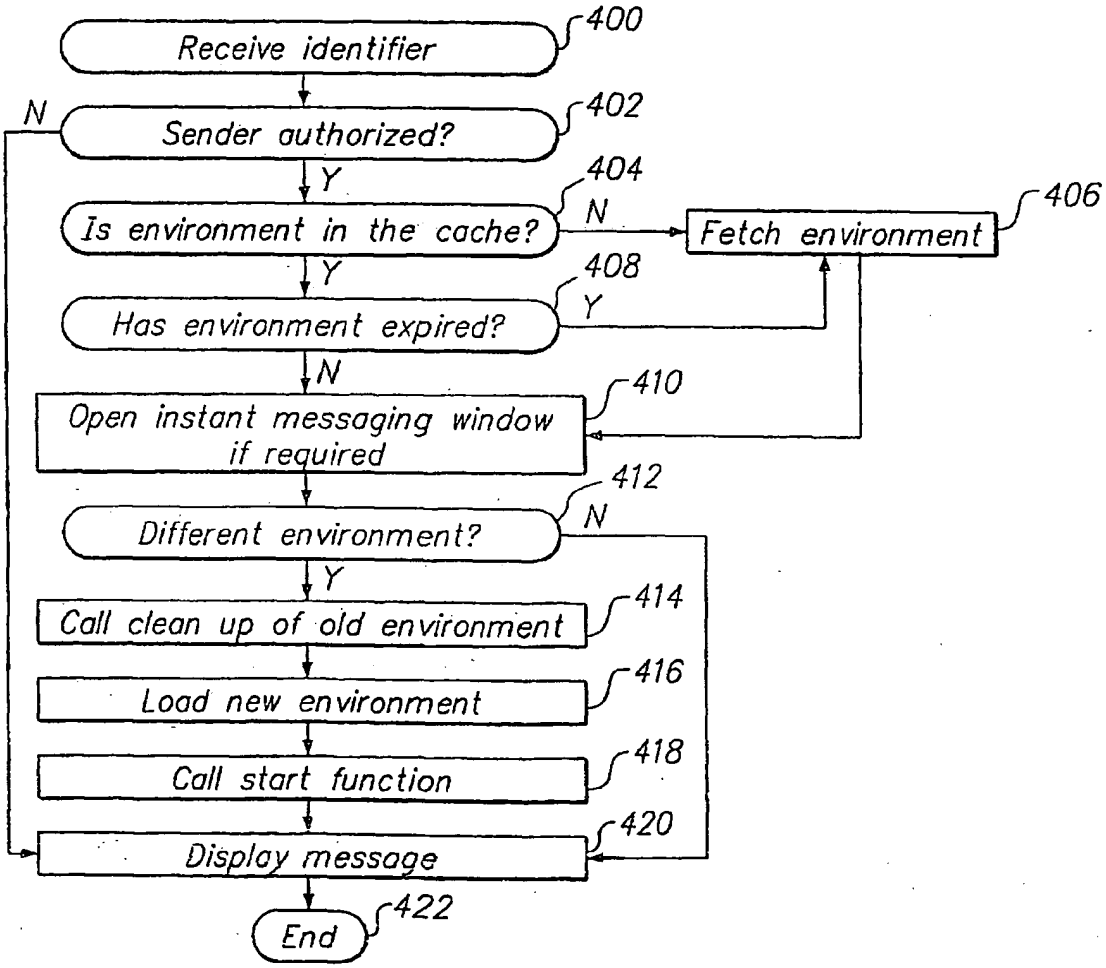

Receive identifier
400
Sender authorized?
402
N
Y
Is environment in the cache?
404
N
Fetch environment
406
Y
Has environment expired?
408
Y
N
Open instant messaging window if required
410
Different environment?
412
N
Y
Call clean up of old environment
414
Load new environment
416
Call start function
418
Display message
420
End
422

FIG. 5

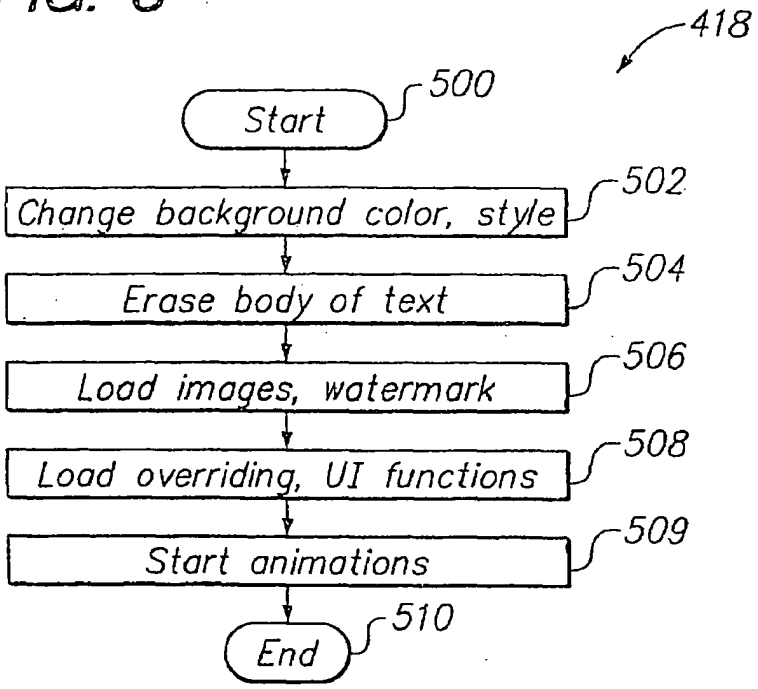

418
500
Start
502
Change background color, style
504
Erase body of text
506
Load images, watermark
508
Load overriding, UI functions
509
Start animations
510
End

FIG. 6

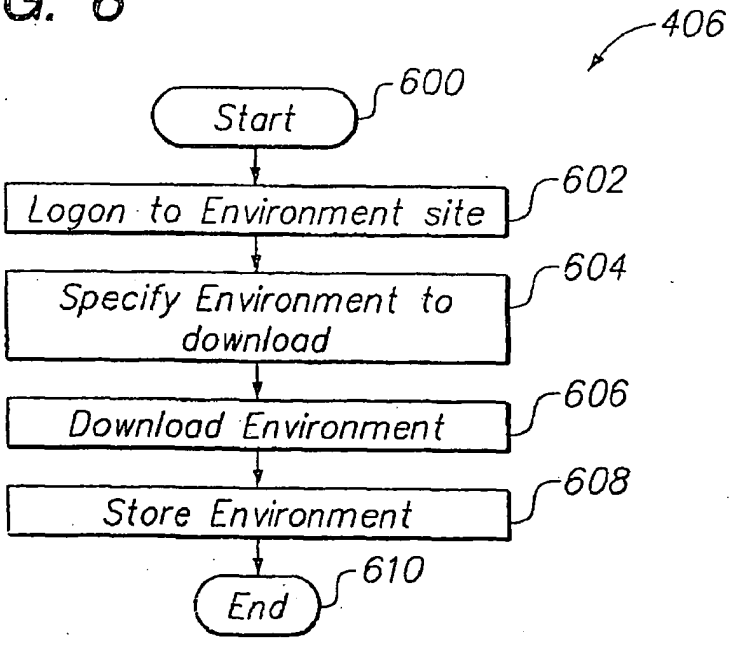

406
600
Start
602
Logon to Environment site
604
Specify Environment to download
606
Download Environment
608
Store Environment
610
End

FIG. 7

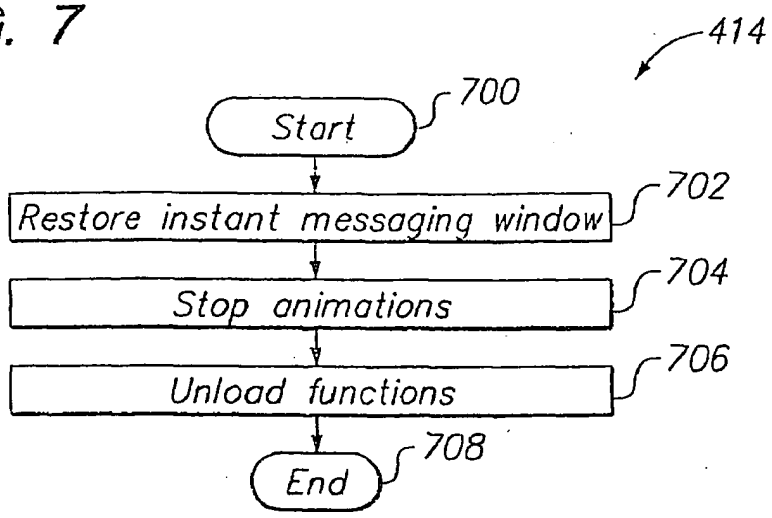

414
Start
700
Restore instant messaging window
702
Stop animations
704
Unload functions
706
End
708

FIG. 8

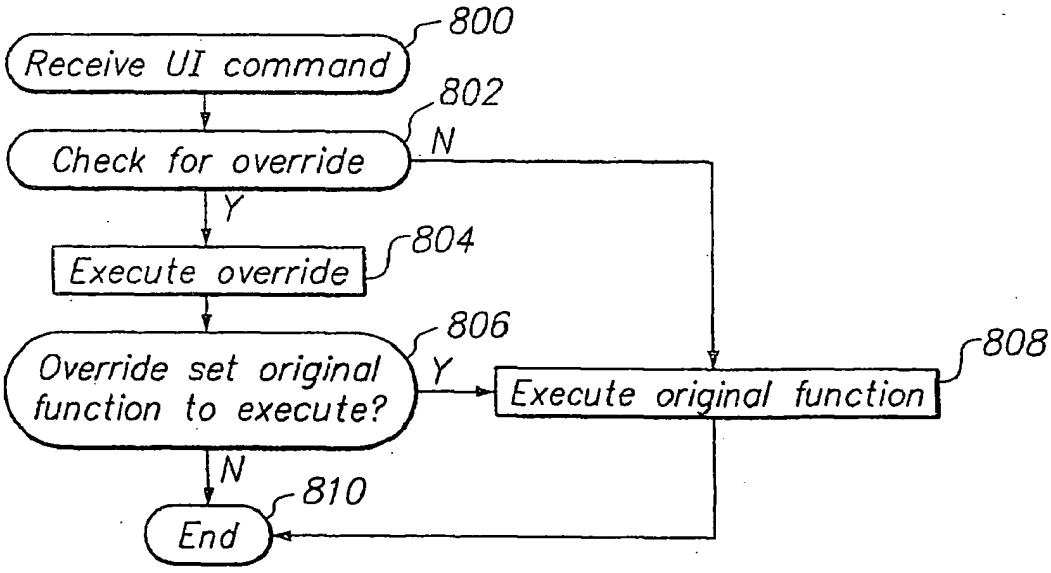

Receive UI command
800
Check for override
802
N
Y
Execute override
804
Override set original function to execute?
806
Y
Execute original function
808
N
End
810

FIG. 9A

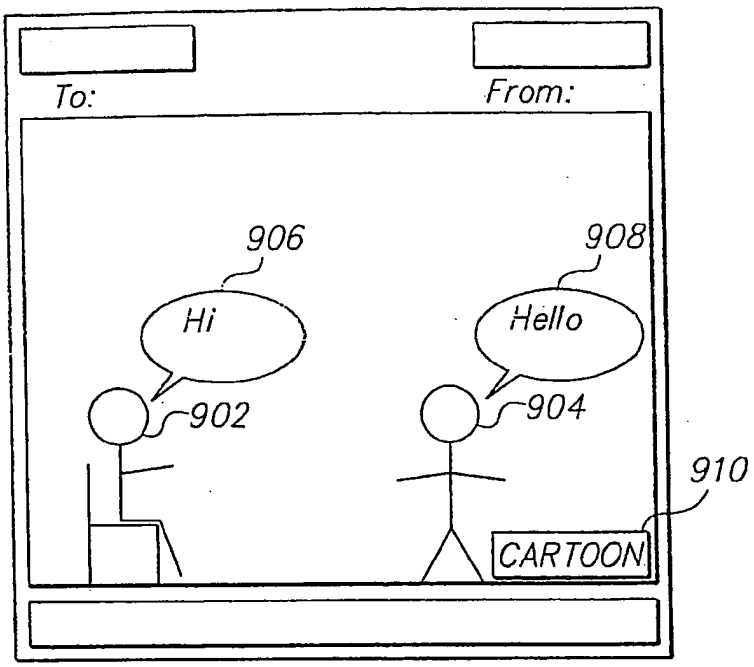

To:
From:
906
Hi
902
908
Hello
904
910
CARTOON

FIG. 9B

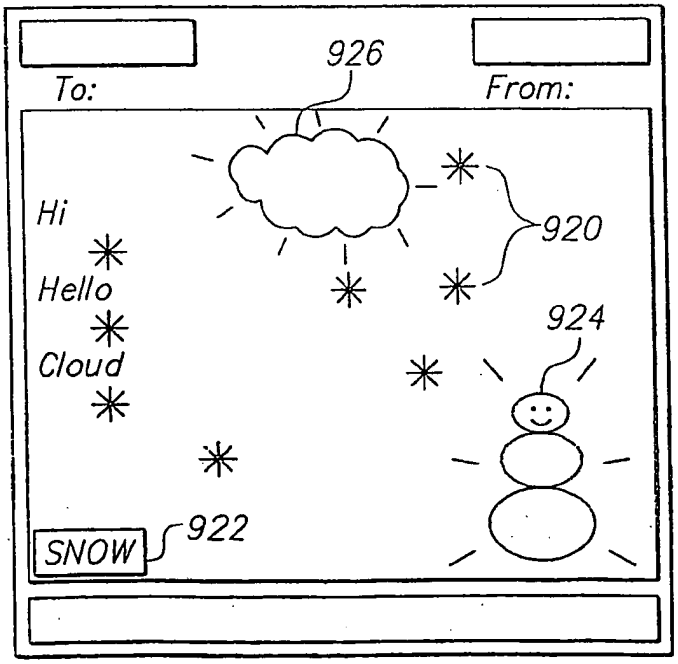
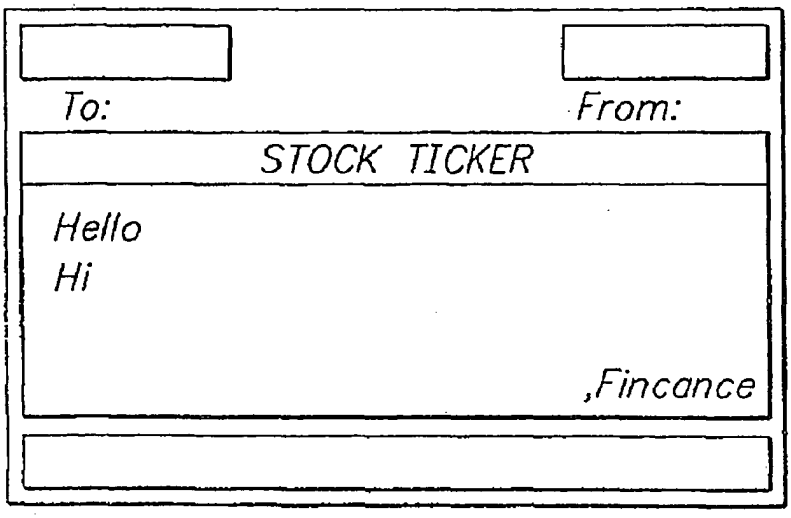

To:
926
From:
Hi
Hello
Cloud
920
924
SNOW
922

FIG. 9C



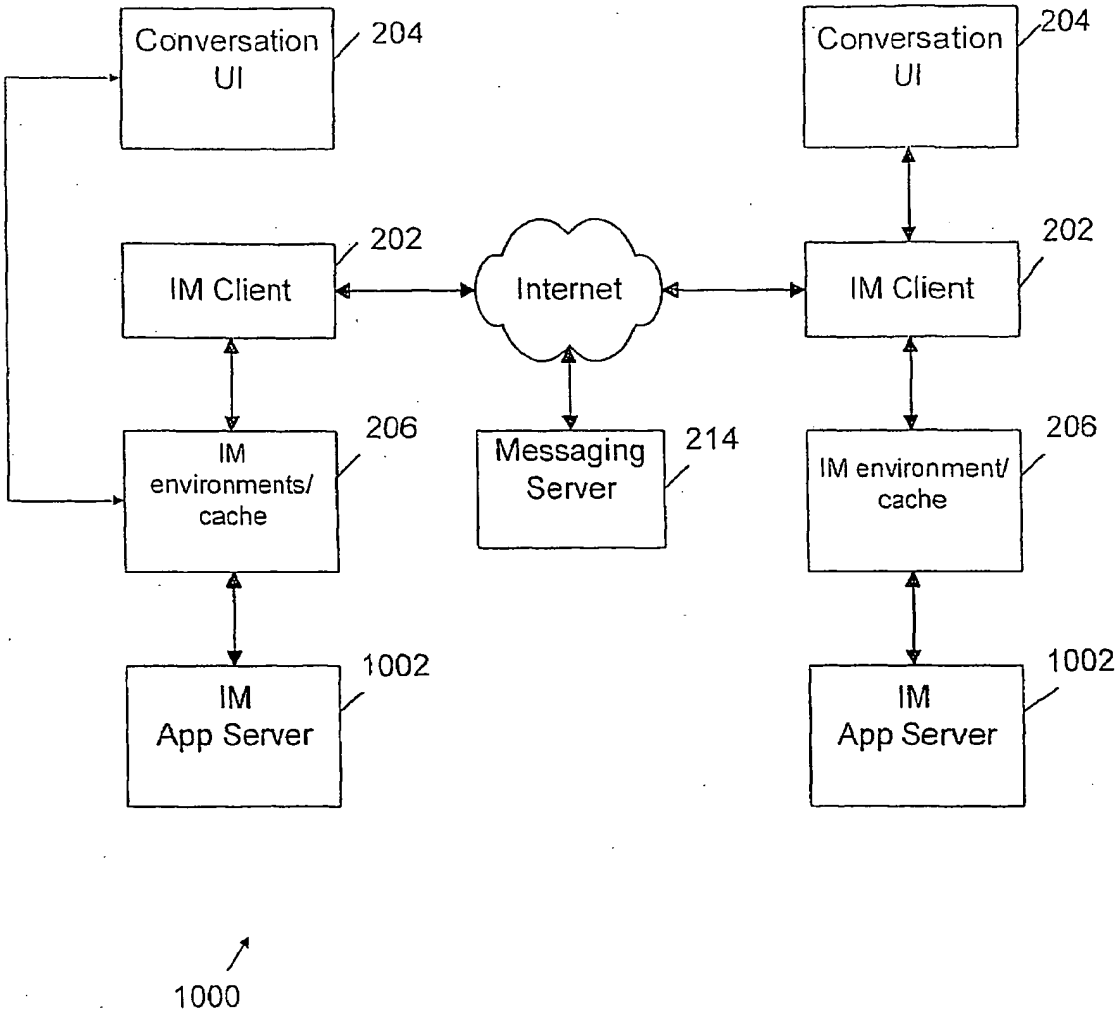

Conversation UI
204
Conversation UI
204
IM Client
202
Internet
IM Client
202
IM environments/ cache
206
Messaging Server
214
IM environment/ cache
206
IM App Server
1002
IM App Server
1002
1000

FIG. 10A

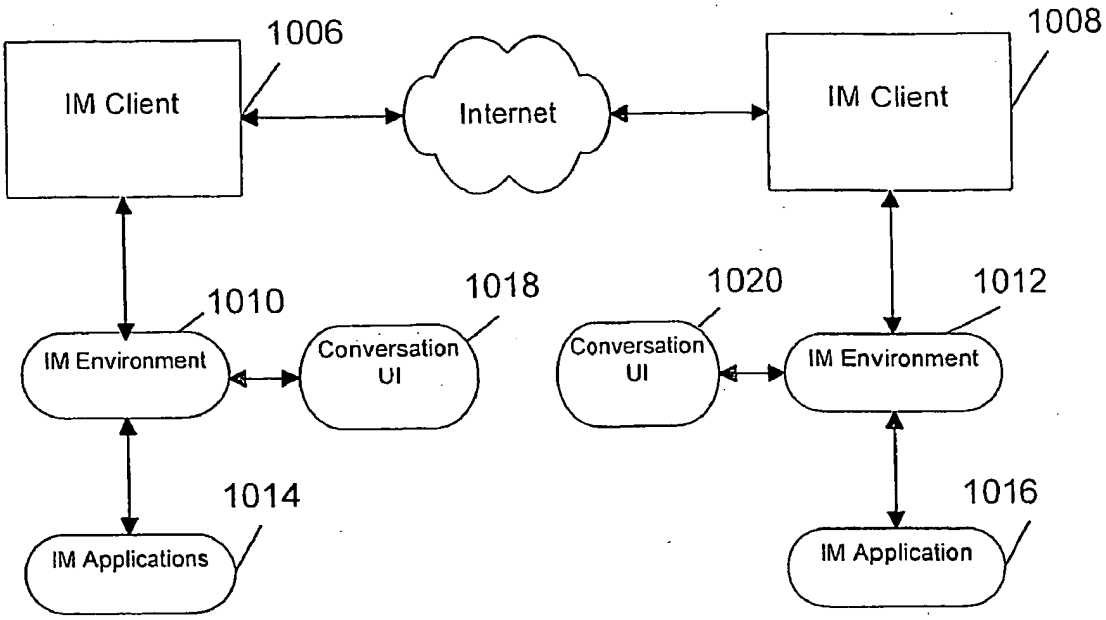

1006
IM Client
Internet
1008
IM Client
1010
IM Environment
1018
Conversation UI
1020
Conversation UI
1012
IM Environment
1014
IM Applications
1016
IM Application

FIG. 10B

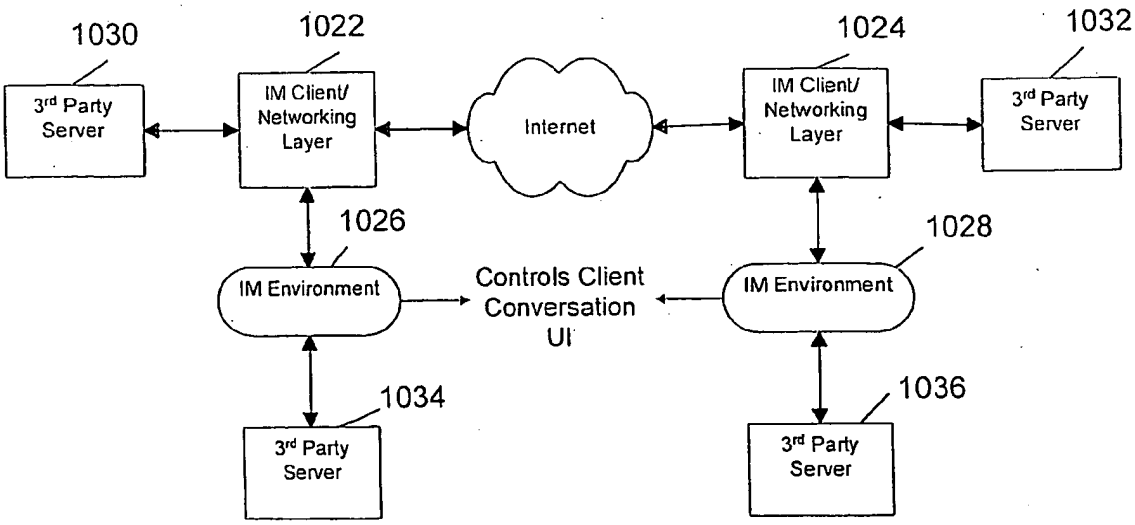

1030
1022
1024
1032
3rd Party Server
IM Client/ Networking Layer
Internet
IM Client/ Networking Layer
3rd Party Server
1026
1028
IM Environment
Controls Client Conversation UI
IM Environment
1034
1036
3rd Party Server
3rd Party Server

FIG. 10C

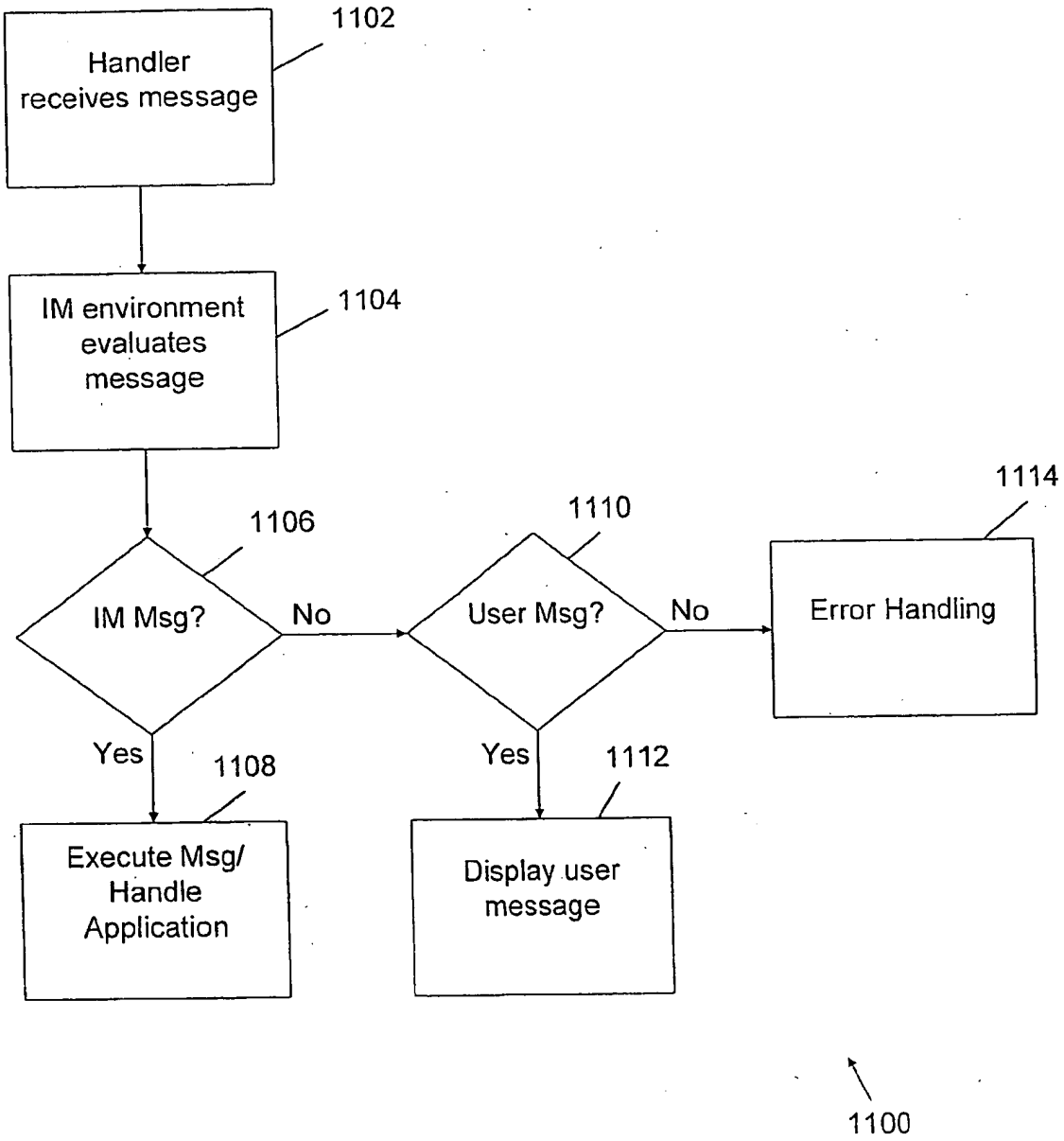

Handler receives message
1102
IM environment evaluates message
1104
IM Msg?
1106
No
User Msg?
1110
No
Error Handling
1114
Yes
1108
Execute Msg/ Handle Application
Yes
1112
Display user message
1100

FIG. 11

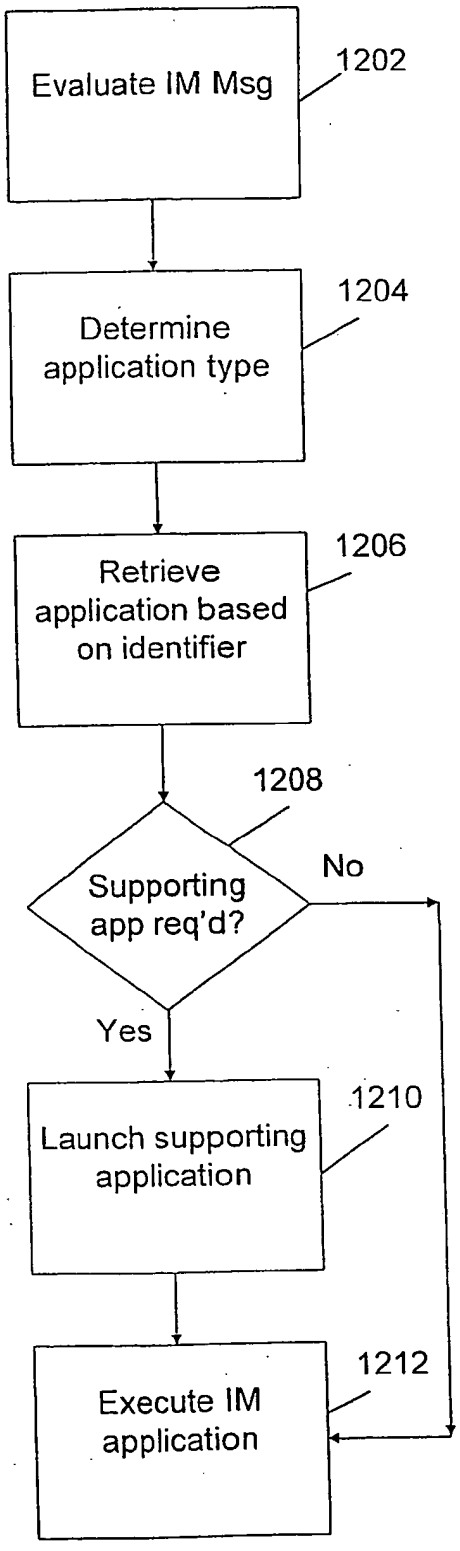

Evaluate IM Msg
1202
Determine application type
1204
Retrieve application based on identifier
1206
1208
Supporting app req'd?
No
Yes
Launch supporting application
1210
Execute IM application
1212
1200

FIG. 12A

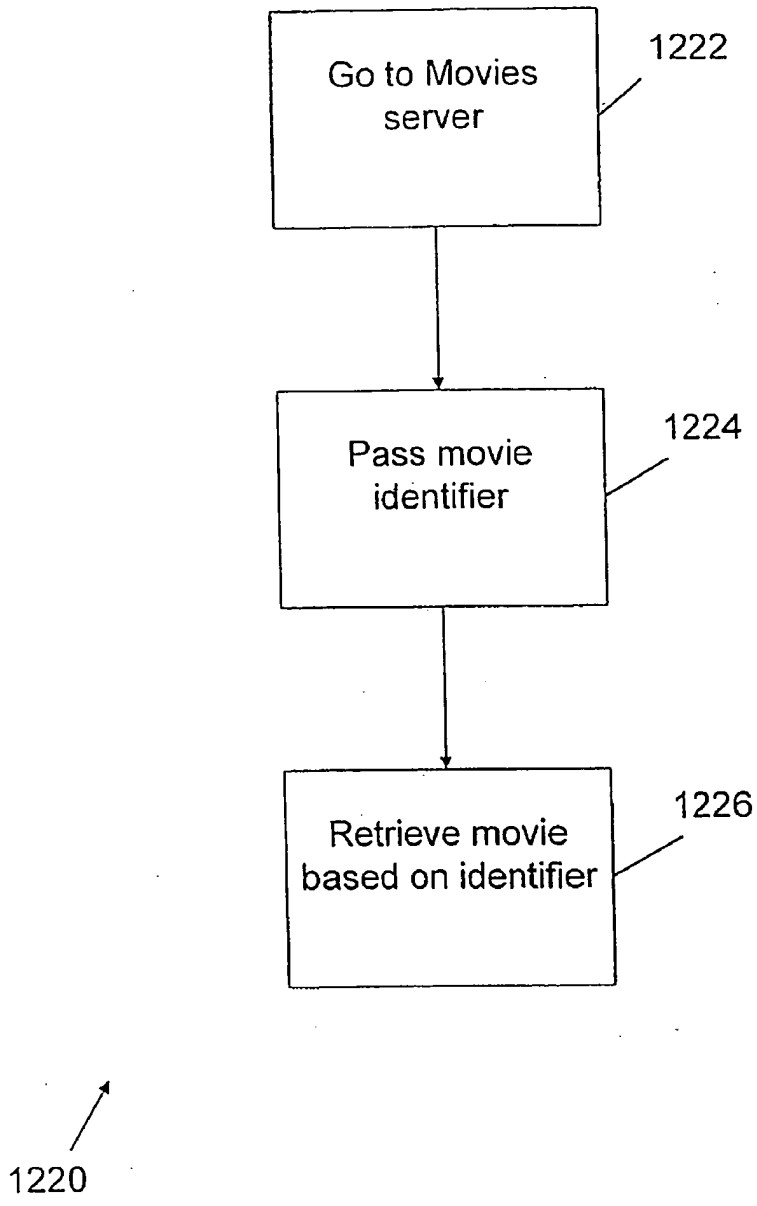

Go to Movies server
1222
Pass movie identifier
1224
Retrieve movie based on identifier
1226
1220

FIG. 12B

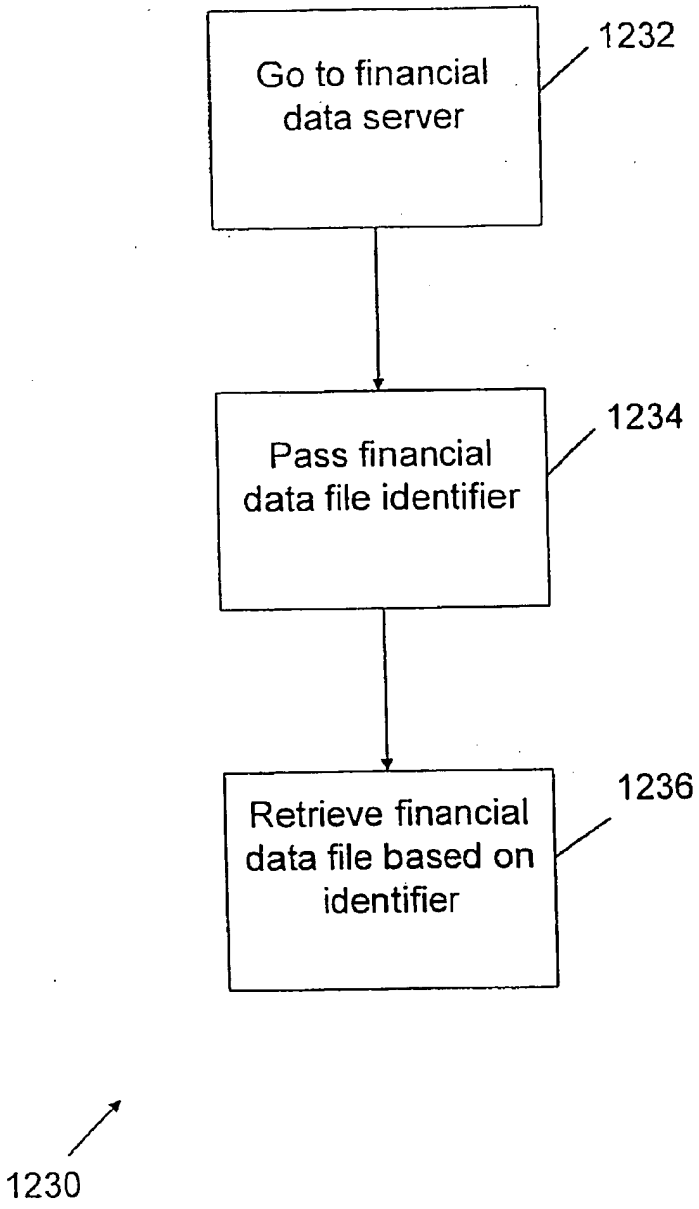

Go to financial data server
1232
Pass financial data file identifier
1234
Retrieve financial data file based on identifier
1236
1230

FIG. 12C

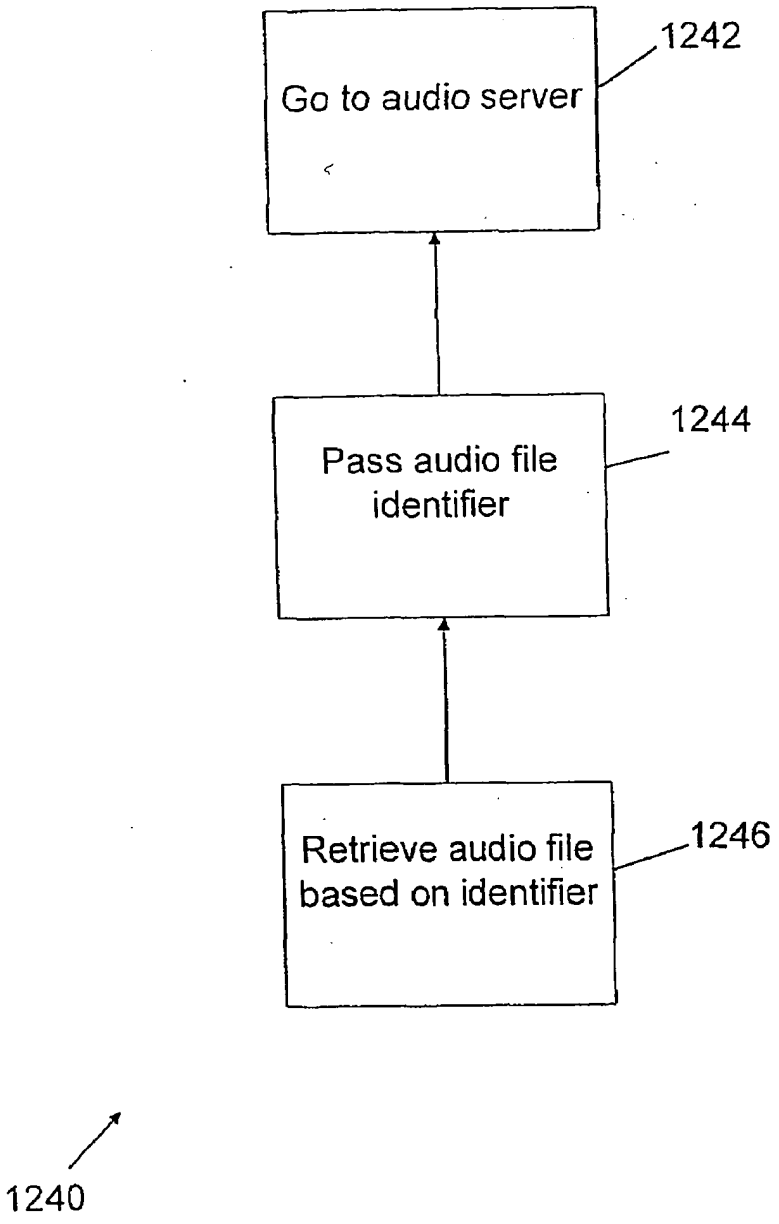

Go to audio server
1242
Pass audio file identifier
1244
Retrieve audio file based on identifier
1246
1240

FIG. 12D

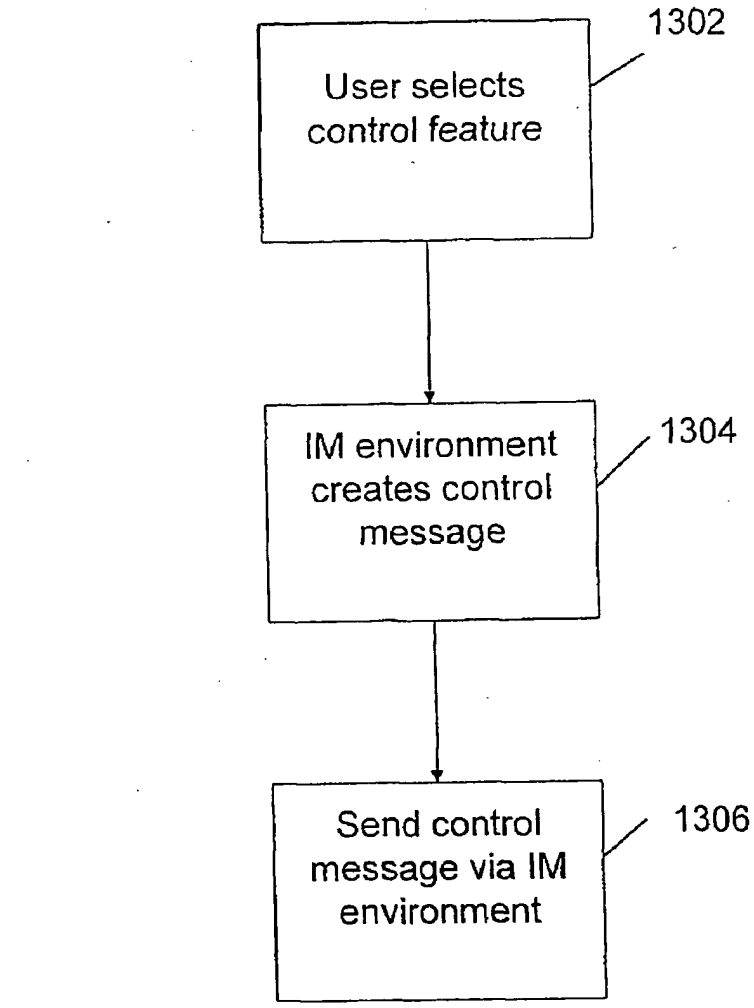

1302
User selects control feature
1304
IM environment creates control message
1306
Send control message via IM environment
1300

FIG. 13

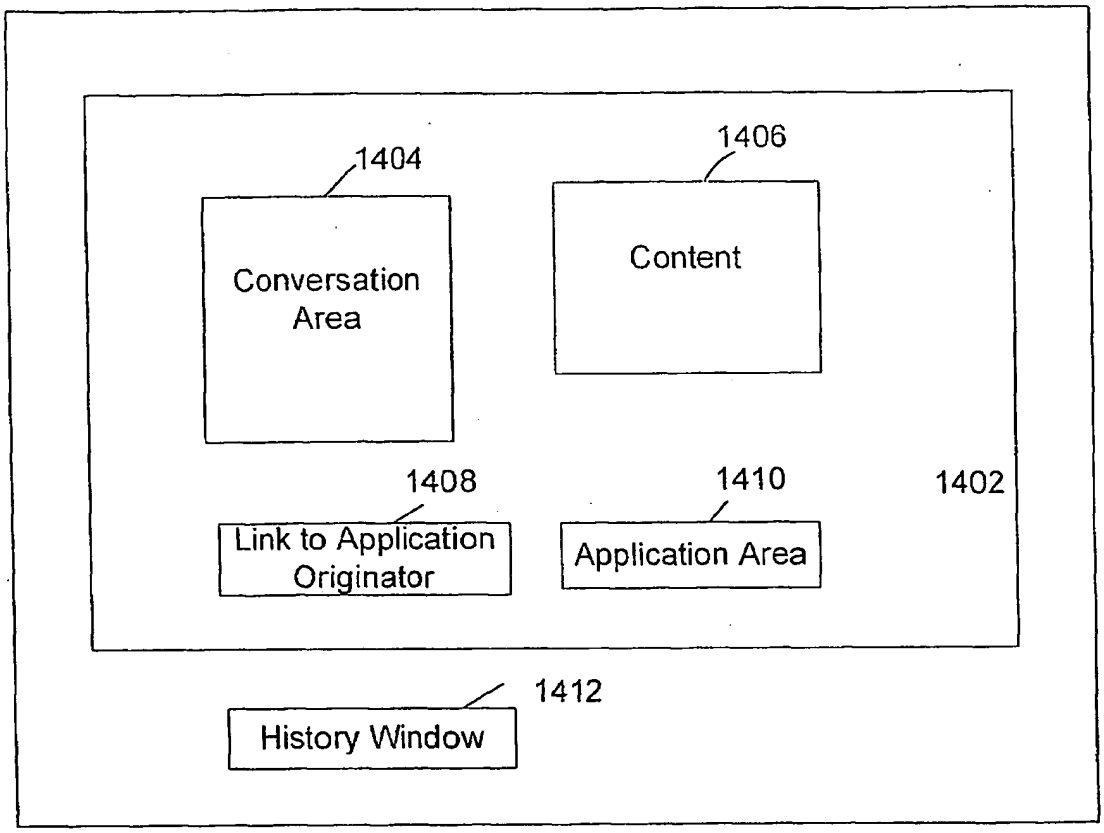

1404
Conversation Area
1406
Content
1408
Link to Application Originator
1410
Application Area
1402
1412
History Window
1400

FIG. 14A

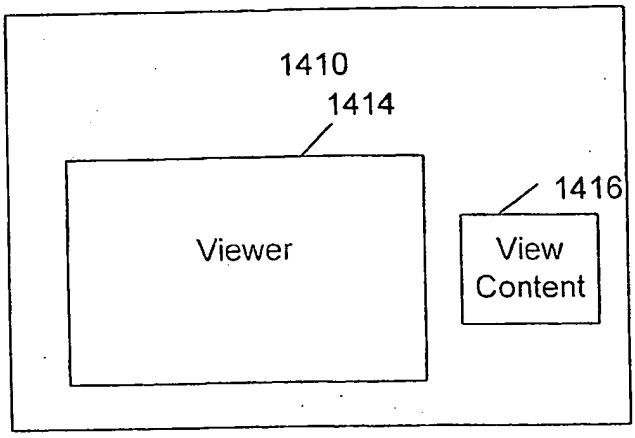


FIG. 14B

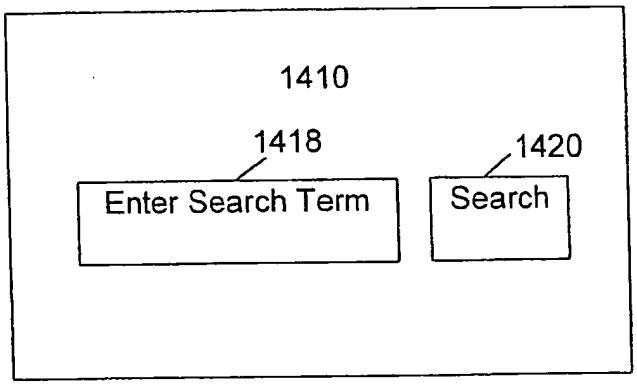


FIG. 14C

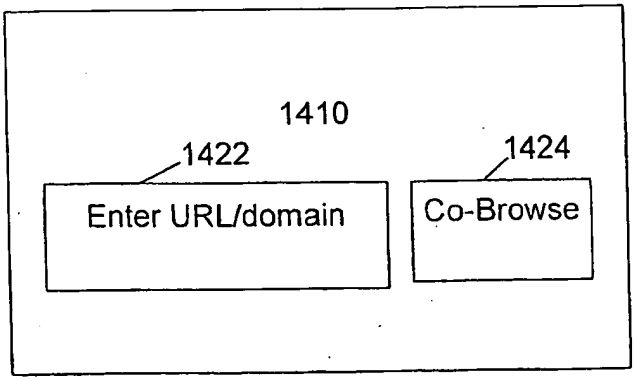


FIG. 14D

MESSENGER-CONTROLLED APPLICATIONS IN AN INSTANT MESSAGING ENVIRONMENT

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation in part of U.S. patent application Ser. No. 09/930,878 entitled "INSTANT MESSAGING ENVIRONMENTS" filed Aug. 15, 2001, now U.S. Pat. No. 7,133,900 which claims priority to U.S. Provisional Patent Application No. 60/331,331 entitled "INSTANT MESSAGING ENVIRONMENTS" filed Jul. 6, 2001, both of which are incorporated herein by reference for all purposes.

FIELD OF THE INVENTION

The present invention relates generally to a data transfer and application server system. More specifically, techniques for messenger-controlled applications in instant messaging environments are disclosed.

BACKGROUND OF THE INVENTION

Instant messaging has become one of the most popular applications on the Internet. Instant messaging programs generally allow users to send and receive messages. The messages are generated and displayed by an instant messaging client on each end and an instant messaging server may perform various functions to facilitate the transfer of messages for communication or conversation.

FIG. **1** is a diagram illustrating an instant messaging system operating over the Internet. An instant messaging client creates an instant messaging window **102** on a computer that generally includes a history window **104** containing messages that have been exchanged in the past, and a new text window **106** for new messages. In addition, the instant messaging client may display various menus and buttons that activate common instant messaging functions such as changing font, ringing another user, inserting symbols, etc. The instant messaging window also generally includes a space where the sender and the recipient of the message are identified.

A message server **112** is also connected to the Internet. In various instant messaging systems, the message server may perform different functions such as receiving messages and transferring them, replacing certain text with symbols, or otherwise modifying or relaying messages. A second instant messenger client also creates an instant messaging window **122** that also includes a history window and a new text window. Instant messaging window **122** displays a message sent from instant messaging window **102** via message server **112**. It should be noted that software intended for implementing an instruction set in an instant messaging environment may be generally referred to as an instant messenger application or IM application.

Each instant messenger client provides for various commands to enable a user to interact with the instant messenger window in various ways, and, to some extent, configure the window to the user's taste. For example, the user may select a color for the history window from among various colors available to the user. In this manner, the user configures his instant messenger window to have a desired appearance. In addition, a user may turn certain features such as ringing, on or off.

The instant messaging client may also be configured to react to certain content, in a limited fashion, received as a message by running a routine or performing some function. For example, when a ring is received, the instant messaging client may be configured to react to the ring by playing a sound, shaking, or executing some other action. The instant messaging client may also be configured not to respond to such a ring. Also, the instant messaging client may be configured to insert an image or play a sound when certain text is included in a message. Thus, a user is generally able to exercise a certain amount of control over the appearance and operation of his instant messaging client. However, users are generally unable to affect the environment or state of another user's instant messaging client or window. Moreover, users are also generally unable to implement, execute, or control instant messaging applications in another user's instant messaging environment. Conventional messaging applications are generally limited to and based upon text contained in user messages exchanged between instant messaging users.

Thus, there are numerous problems which exist with regard to conventional techniques. Conventionally, users are unable to execute applications beyond an instant messaging environment to include other users engaged in the same instant messaging session. Users are generally only able to send messages and configure their own instant messaging applications to perform certain actions limited to the instant message content. Moreover, users are generally unable to configure, execute, and control applications in an instant messaging environment between two or more users. It would be useful if a reliable system and method could be provided for instant messaging users to execute and control applications in an instant messaging environment.

SUMMARY OF THE INVENTION

A system and method for messenger-controlled applications in an instant messaging environment is disclosed. An instant messaging environment may affect the execution and operation of an application in an instant messaging session. A sending instant messaging environment can specify an instant messaging environment identifier along with an application identifier and cause a receiving instant messaging application to implement a selected environment. The selected instant messaging environment may affect the state of the instant messaging application in a number of ways. The environment may affect how user interface commands are processed as well as the appearance of the instant messaging window and the manner in which information is sent and processed.

The environment may also determine the type of instant messaging application to be executed as well as an associated mode of execution. In some embodiments, an instant messaging application may be configured to automatically, semi-automatically, or manually execute. In other embodiments, instant messaging applications may require the use of a supporting application for execution. Instant messaging environments may be downloaded from one or more central instant messaging servers so that users may obtain new environments to specify to other users. When the instant messaging application receives an environment identifier for which the instant messaging application does not have the corresponding environment, that instant messaging application may also download the environment from the central instant messaging server. Likewise, when an instant messaging environment receives an identifier for an instant messaging application, the instant messaging environment

can request and retrieve the selected instant messaging application based upon the application identifier.

It should be appreciated that the present invention can be implemented in numerous ways, including as a process, an apparatus, a system, a device, a method, or a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. Several inventive embodiments of the present invention are described below.

In one embodiment, a method is disclosed for selecting, at a first client, the application in the instant messaging environment, configuring an instant messaging control message for the application, including an identifier related to the application selected at the first client, sending the instant messaging control message to a second client, and executing the application in the instant messaging environment using the control message to retrieve the application from a server, unless the application has been previously called by the user.

In another embodiment, a method is disclosed for selecting a control feature in a first instant messaging environment, creating a control message based on selecting the control feature, and sending a control message from the instant messaging environment to a second instant messaging environment.

In yet another embodiment, a system is disclosed for an instant messaging environment for creating a control message and a messenger for sending or receiving the message and selecting an instant messaging environment in which to execute the application.

These and other features and advantages of the present invention will be presented in more detail in the following detailed description and the accompanying figures which illustrate by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

FIG. **1** is a diagram illustrating an instant messaging system operating over the Internet;

FIG. **2** is a block diagram illustrating an instant messaging system that implements instant messaging environments stored by instant messaging applications;

FIG. **3** is a diagram illustrating how an instant messaging environment is implemented using a browser control window;

FIG. **4** is a flow chart illustrating a process executed by an instant messaging application when an environment identifier is received;

FIG. **5** is a flow chart illustrating in further detail a process that is executed when the start function is called in step **418** of FIG. **4**;

FIG. **6** is a flow chart illustrating a process implemented to load an environment as shown in step **406** of FIG. **4**;

FIG. **7** is a flow chart illustrating a process implemented to clean up an old environment;

FIG. **8** is a flow chart illustrating a process executed by the instant messaging application in response to a user interface command;

FIG. **9A** is a diagram illustrating a cartoon instant messaging environment such as the one described above;

FIG. **9B** is a diagram illustrating an environment where a snow theme has been implemented;

FIG. **9C** is a diagram illustrating another environment where the ability of an environment to interact with another application is illustrated;

FIG. **10A** is a diagram illustrating an exemplary instant messaging system that implements messenger-controlled applications;

FIG. **10C** is a diagram illustrating another embodiment of an instant messaging system using third-party servers;

FIG. **10B** is a diagram illustrating an alternative embodiment of an instant messaging system;

FIG. **11** is a flow chart illustrating an exemplary process **1100** for handling message-borne applications in an IM environment;

FIG. **12A** is a flow chart illustrating the control and execution of IM applications in an instant messaging system, in accordance with one embodiment of the present invention;

In FIG. **12B**, an exemplary flow chart for implementing a messenger-controlled IM application such as a movie trailer is shown;

FIG. **12C** illustrates another exemplary flow chart for implementing a messenger-controlled IM application which yields financial data such as a stock quote, earnings indicator, or other financial data;

FIG. **12D** illustrates another exemplary flow chart for implementing a messenger-controlled IM application such as an audio file (e.g., song, recording, etc.);

FIG. **13** is a flow chart illustrating an exemplary IM application control process in accordance with one embodiment of the present invention;

FIG. **14A** is an exemplary user interface for controlling IM applications in accordance with one embodiment of the present invention;

FIG. **14B** illustrates another exemplary embodiment of application area **1410** for viewing content;

FIG. **14C** illustrates another exemplary embodiment of application area **1410** for entering a search term and performing a search based on the search term; and

FIG. **14D** illustrates another exemplary embodiment of application area **1410** for executing search functionality such as co-browsing, in an IM environment.

DETAILED DESCRIPTION

A detailed description of a preferred embodiment of the invention is provided below. While the invention is described in conjunction with that preferred embodiment, it should be understood that the invention is not limited to any one embodiment. On the contrary, the scope of the invention is limited only by the appended claims and the invention encompasses numerous alternatives, modifications and equivalents. For the purpose of example, numerous specific details are set forth in the following description in order to provide a thorough understanding of the present invention. The present invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the present invention is not unnecessarily obscured.

An instant messaging environment is a shared environment which exists between 2 or more instant messaging users. FIG. **2** is a block diagram illustrating an instant messaging system that implements instant messaging environments stored by instant messaging applications. It should be noted that, while this description refers extensively to instant messaging systems, the disclosed environments are

also applied to other messaging systems in different embodiments. In general, an instant messaging system refers to any real time or near real time messaging or information exchange system. Many such systems will buffer messages sent to a participant who is temporarily unavailable or offline. A notification may be sent to the participant. An instant messaging environment may be specified by one instant messaging application to change the environment or state of another instant messaging application. A first instant messaging client, **202** interacts with a conversation user interface **204** that displays information to and receives messages from a user.

It should be noted that the term "application" as used herein is intended to refer to any client application, server application, distributed application, self contained application or combination thereof. An application may be implemented in any appropriate manner, including being embedded in a chip or being loaded into memory or firmware.

In some embodiments, the conversation user interface includes a conventional instant messaging window as shown in FIG. **1** with a history window for displaying previous messages and a new message window for composing new messages. In other embodiments, the conversation user interface may be dramatically different. For example, in one embodiment, the conversation user interface of the history window is modified to resemble a cartoon interface that shows the participants in instant messaging as comic characters or avatars having a conversation with text bubbles used to list messages back and forth. The conversation user interface may also include multiple history windows and other menus associated with instant messaging features such as buddy lists, formatting options, etc. Instant messaging environments as described herein may generally be applied to any type of conversation user interface used to display and author messages.

User interface commands are transferred from the instant messenger client to the conversation user interface to cause the conversation user interface to display instant messages and their corresponding environment to the user. For example, an append command is used to send a new message received from another instant messaging application to the conversation user interface. The append command may simply cause the received message to be added to a history window if a generic environment is selected. It should be noted that the phrase "user interface commands" as used herein is intended to encompass any functions, behaviors, actions, capabilities, etc. that are features of the user interface or the instant messaging window.

Other instant messaging environments may cause the append command to behave in different manners. For example, in the cartoon instant messaging environment, the append command would cause the last text bubble generated for the character corresponding to the message sender to disappear and for a new text bubble containing the current message to be created. Thus, the selected environment affects the state of the instant messaging application so that a user interface command specified by the instant messaging client may have different effects depending on the selected environment. A number of different environments may be stored in a cache **206**.

In discussing the example shown, for the purpose of explanation, instant messaging client **202** will be described as sending a message to instant messaging client **212**. In general, the two instant messaging clients will both send and receive messages in turn. Instant messaging client **202** sends a message that includes an environment identifier along with the message over the Internet. Throughout this specification, the Internet is referred to as a medium over which messages are sent. The disclosed system also operates over any other appropriate network or internetwork including wireless networks, proprietary networks, intranets, local area networks, or wide area networks. In one embodiment, instant messaging server **214** receives a message, processes the message and transfers the message to instant messaging client **212**. In other embodiments, a peer to peer messaging system is implemented in which an instant messaging server need not necessarily become involved in transferring messages between instant messaging clients.

If a messaging server does process messages, then the messaging server may perform certain checks to determine whether the environment identifier as specified along with the message corresponds to a valid environment. In some embodiments, environments may expire as a result of a sponsorship or other type of agreement to maintain the environment lapsing.

The message along with the environment identifier is received by instant messaging client **212**. Instant messaging client **212** also stores in a cache **220** various environments that it has obtained in the past. Instant messaging client **212** searches the cache for an environment that corresponds to the environment identifier received from instant messaging client **202**. If a corresponding environment is found, then instant messaging client **212** changes the environment currently implemented by it.

The current environment affects how user interface commands sent from instant messaging client **212** to conversation user interface **216** are processed. In one embodiment, the instant messaging client may check whether or not an environment has expired before it implements that environment. If an environment has expired, then the instant messaging client may retrieve an updated environment from the instant messaging server or another specified source via the network.

FIG. **3** is a diagram illustrating how an instant messaging environment is implemented using a browser control window. It should be noted that a browser control window as used herein is intended to refer to any program control or set of controls that can interpret and render scripted pages. Also, any other appropriate system for rendering information may be used. Instant messaging window **300** is configured in a similar manner as a conventional instant messaging window with a TO field, a FROM field, and various function and menu buttons surrounding a history window **302**. However, history window **302** is implemented in an unconventional manner. History window **302** is implemented using code in a browser control that processes an HTML file which contains the formatting for the history window and controls the appearance of the instant messaging cache that appears therein.

In one embodiment, the instant messaging environment is implemented by loading into memory one or more JavaScript objects that implement methods and behaviors that override the history window's default methods and behaviors. Of course, objects may be written in any appropriate programming language or system, such as ActiveX. In another embodiment, an Iframe that is not visible to the user includes JavaScript that implements the instant messaging environment. For example, if a message is received by the instant messaging client and an append function is called to add that message to the history window, then the environment embedded in the JavaScript code alters the way the browser control processes the append function. For example, the JavaScript may include a different version of the append function that causes text to be appended in a different

manner than it would normally be appended if no environment were implemented by the code in the browser control. It should be noted that loading a JavaScript object without using an Iframe is generally preferred over the Iframe implementation for the sake of speed. This description refers to both implementations and it should be noted that when one implementation is referenced, that the other implementation (or any reasonable alternative implementation) may be used in different embodiments.

Once the code in the browser control has executed the modified append function contained in the environment, the environment may either allow the normal append function to be subsequently executed or may indicate that the normal append function should not be executed, essentially replacing the normal append function with the modified version. For example, an environment may cause a sound or visual effect to occur every time a new line is appended to the history window. In such a case, the environment would cause the browser control upon receiving the append function to first execute some JavaScript that would play the sound or create the visual effect and then return to the normal append function for the text to be appended in a conventional manner. In another example, such as the one described above where the history window includes cartoon characters and text bubbles that contain the appended text, the normal append function would be replaced by the append function contained in the environment and the normal append function would not be called upon completion of the executed environment code.

Thus, implementing code in a browser control within an instant messaging window that renders an HTML allows user interface commands to be redefined. Different environments are stored simply as different sets of code. The JavaScript contained in the environment redefines certain commands or functions by executing additional commands and then either calling or not calling the original function as desired.

It is important to note that, while the above described JavaScript code implementation of instant messaging environments has significant advantages, it is by no means the only manner in which instant messaging environments are implemented. In different embodiments, instant messaging environments are implemented by using different code structures. In general, a certain set of code representing the environment is accessed by the instant messaging application. The set of code is loaded in response to the specification of an instant messaging environment identifier by another instant messaging application. The environment may also be specified by the user of the application. By way of example and without limitation, instant messaging environments may be implemented using the wireless application protocol, XML, VRML, or any other appropriate public or private standard.

The specified environment alters the response of the instant messaging application to actions performed by the user. As shown in FIG. **2**, in one embodiment, this alteration occurs by redefining commands or functions sent from the instant messaging client to the conversation user interface. As explained further in FIG. **3**, in one embodiment, this may be implemented by using a browser control to render a history window and loading a JavaScript object that includes the function definitions. It should be apparent that many alternatives exist for implementing this system in a similar manner.

As a further example, another function that may be redefined is the scroll function. The user may indicate by dragging the mouse over an arrow button in the instant messaging window or by some other convenient means that he would like to scroll the history window. When such a scroll command is received, then a check is first made to determine whether or not the scroll command has been redefined by a loaded environment. For example, an environment may contain a watermark that is intended to always show up in the history window. In such a case, the scroll function would be redefined so that the watermark does not scroll within the window while the rest of the text inside the window does scroll.

As shown in FIG. **2**, environment identifiers are passed between instant messaging applications by including an environment identifier along with each message. In one embodiment, there is a default environment identifier that is automatically passed between instant messaging applications if no environment has been specified by either of the users. Once one of the users specifies an environment, if the other user accepts the environment then that user will then pass the accepted environment identifier back to the user that originally specified the environment identifier, thus maintaining the environment. Environments may be selected by users in a variety of ways. For example, audio or video user inputs may be processed or used to select or modify an environment. Also, a robot or other program communicating with a user may select or modify an environment. Other context information about the user's system, a specified default environment or set of environments, or message content may select or modify an environment. One environment may select or modify another environment and a central instant messaging server may itself specify or modify an environment.

If an instant messaging application passes an environment identifier that is not accepted by the receiving instant messaging application, then the receiving instant messaging application will send back the default environment identifier and the sending instant messaging application may either change back to the default environment or maintain its own environment according to how it is configured. Thus, the instant messaging environment is either maintained or changed with each message passed back and forth containing an environment identifier.

FIG. **4** is a flow chart illustrating a process executed by an instant messaging application when an environment identifier is received. The environment identifier is received in step **400**. Next, in step **402**, it is determined whether the sender of the identifier is an authorized sender. In one embodiment, this is determined by whether the sender is a member of a buddy list or other list maintained by the receiver of parties eligible to send environment identifiers and change the environment of the recipient's instant messaging application. If the sender is not authorized, then control is transferred to step **420** and the message is displayed.

If the sender is authorized, then control is transferred to step **404** and it is determined whether the environment is already cached by the receiving instant messaging application. If the environment is not cached by the receiving instant messaging application, then control is transferred to a step **406** where the environment is fetched. In one embodiment, the environment is loaded by accessing a website that makes environments available for download. The environment identifier may be sent to the website to identify the environment to be downloaded. In other embodiments, the environment may be obtained from the sender and an authentication code or signature may be used to verify the integrity of the environment. Once the environment is downloaded, then control is transferred to step **410**. If the envi-

ronment is in the cache, then control is transferred from step **404** to step **408** and it is determined whether the environment has expired.

In one embodiment, environments that are not found in a local cache are obtained from a secure source, such as a trusted website. This secure mode of distribution prevents parties from using environments for hostile purposes. Since instant messaging participants merely refer to environments and the referred to environments are separately obtained from a trusted source, participants do not intentionally or inadvertently send damaging environments to each other.

In one embodiment, whenever an environment is downloaded by an instant messaging application, an expiration date is also provided to the instant messaging application so that the instant messaging application can determine if the environment has expired. In another embodiment, the instant messaging application is required to always check with an instant messaging server to determine whether an environment has expired before that environment is implemented.

If the environment has expired, then control is transferred to step **406** and the environment is loaded. If the environment has not expired, then control is transferred to step **410** and an instant messaging window is opened if an instant messaging window is currently not open. Control is then transferred to step **412** where it is determined whether a different environment has been specified by the received environment identifier. If a different environment has not been specified, then the currently loaded environment may continue to be used and control is transferred to step **420** where the message is displayed.

If the environment is different, then control is transferred to step **414** and a clean up function is called for the old environment. Control is then transferred to step **416** and the new environment is loaded into memory. Next, in step **418**, the start function is called for the new environment so that any code that should be executed when the new environment is loaded is executed. Control is then transferred to step **420** and the message is displayed. The process ends at **422**.

In one embodiment, an environment is downloaded from a website by receiving a series of different types of files. The first type of file is an initialization file that contains information regarding the environment that the instant messenger application can access without actually loading the environment. The initialization file may contain items such as a display name that determines how the environment will be listed in a formatting tool bar used to select environments and an expiration that can be used by the instant messaging application to determine that the environment is invalid and initiate a download of an updated version of the environment.

The second type of file is the actual environment program file that contains, in one embodiment, encoded JavaScript that defines how the environment looks and behaves. The third type of file generally downloaded is media file such as an image file, video clip, animation, sound clip, etc. that provides images or other media that are generally used by the environment. In different embodiments, the different types of files are encrypted or compressed either together or separately and encapsulated or signed as is deemed appropriate. The files may be sent together using the .zip or the .cab or other archive or compression format, if desired. Preferably, at least the JavaScript objects are encrypted.

FIG. **5** is a flow chart illustrating in further detail a process that is executed when the start function is called in step **418** of FIG. **4**. The process begins at **500**. In step **502**, the background, color and style of the instant messaging window is changed to the background, color and style of the environment that has been selected. In step **504**, the body of text is erased. It should be noted that in some environments, the body of text may not be erased but in other environments the body of text may need to be erased. For example, in the cartoon environment shown above, only the current message being conveyed is shown in a text bubble. Next, in step **506**, any images and watermarks associated with the environment are loaded. Then, in step **508**, any overriding user interface functions are loaded so that when a user interface command is received, the function called by the command may be changed to the function specified by the instant messaging environment. In step **509**, any animations (such as falling snowflakes as shown below) associated with the environment are started. The process ends at **510**.

FIG. **6** is a flow chart illustrating a process implemented to load an environment as shown in step **406** of FIG. **4**. The process starts at **600**. In a step **602**, the application logs on to a website that includes environments available for download. In step **604**, the application specifies an environment to be downloaded. In one embodiment, the same environment identifier used by applications to identify environments to each other is used to identify an environment to download. In a step **606**, the environment is downloaded. The environment is stored in step **608**. The process ends at **610**.

Requiring that the instant messaging environment be downloaded from one or more central sites enables editorial control to be exercised over the environments that are sent to users. Obscene environments or other undesirable environments can be avoided. Also, viruses and other malicious code can be excluded. In certain embodiments, users may be allowed to create environments and send them to other users, but it is an important feature that in some embodiments, there is at least one mode where environments are only downloaded from a trusted or controlled source that ensures security and, if desired, some control over content. One important use of instant messaging environments is advertising. When an instant messaging environment includes an ad, controlling the source of the environment facilitates tracking of the environment use as well as making sure that the environment is not altered.

It should be noted that when the environment is downloaded, data in the initialization file for the environment may be separately stored in a table or system registry so that information about loaded environments may be readily accessed by the instant messaging application without opening files. For example, if an expiration date is associated with the environment, then that date may be written into an expiration table so that the instant messaging application can periodically perform clean up and delete expired environments or check before an environment is accessed whether or not it has expired. Likewise, the names of various download environments may be stored in a single table used to create a list of environments for a menu that facilitates the selection of environments by a user.

FIG. **7** is a flow chart illustrating a process implemented to clean up an old environment. The process starts at **700**. In a step **702**, the instant messaging window is restored to a default state. That may involve, for example deleting a watermark created by the old environment. The default state may include a default background color, font, style, etc. In step **704**, any animations or other routines are stopped. In a step **706** redefined functions that replace user interface commands are unloaded. The process then ends at **708**. It should be noted that other clean up operations may typically be included in a clean up process.

FIG. **8** is a flow chart illustrating a process executed by the instant messaging application in response to a user interface

command. The process starts at **800** when the user interface command is received. In step **802**, it is determined whether an overriding or modified function has been defined by the currently loaded environment. If no overriding function is defined, then control is transferred to a step **808** and the original function is executed. If an overriding or modified function does exist, then control is transferred to step **804** and the overriding function is executed. Control is then transferred to a step **806** where it is determined whether the overriding function has set the original function to execute once it has been completed. If the original function is to execute, then control is transferred to step **808** and the original function is executed. If the original function is not to be executed, then control is transferred to step **810** and the process ends.

Sending environment identifiers between instant messaging applications and using the environment identifiers to change the environment of a receiving application has been described. Once this capability of redefining user interface command functions, executing animations and creating event driven actions to occur in an instant messaging window has been enabled, a rich variety of environments can be defined for instant messaging applications. Some examples of instant messaging environments that may be created are provided in FIGS. **9**A through **9**C.

FIG. **9**A is a diagram illustrating a cartoon instant messaging environment such as the one described above. Characters **902** and **904** represent participants in the instant messaging session and text bubbles **906** and **908** display the last message sent by each of the participants. This environment is implemented by redefining the append function to delete an existing text bubble above the character representing the sender of the text message and then insert the message being sent into a new text bubble. In addition, the start function of the environment deletes all text in the history window and draws in the characters that are part of the cartoon. In addition, a watermark **910** is provided that shows the name of the cartoon that is the theme of the environment. In many embodiments, the watermark also functions as a link so that a user may link to a website related to the environment. This technique is especially useful when the environment is part of an ad campaign.

The instant messaging server may track statistics about environments for various reasons. For example, if an environment has an advertisement theme associated with it, then each time the environment is specified by a user to another user that event may be counted by the server so that ad revenue can be generated. In addition, when certain functions within the environment are called as a result of events, the instant messaging server may record such occurrences as well and those events may also generate ad revenue.

FIG. **9**B is a diagram illustrating an environment where a snow theme has been implemented. Animated snowflakes **920** fall down the window so long as the environment is active. In addition, a watermark **922** is present that provides a link to a site related to snow. Also, when the ring or buzzer function is called by one of the users, then instead of the general ringing, a snowman **924** appears and flashes. Also, when certain text is typed in the window such as the word cloud, a cloud **926** appears.

FIG. **9**C is a diagram illustrating another environment where the ability of an environment to interact with another application is illustrated. In the environment shown, a stock ticker obtains information from either the instant messaging program, a browser, or other program that makes stock information available to user. In addition, the environment includes a finance watermark link that directs the user to a financial site. In this environment, when the environment is loaded, a function is periodically executed that checks for data from the stock ticker application. In addition, environments can communicate information to other environments by sending messages. Also, environments can read certain information available to a messenger program such as a buddy list or in some cases stock quote information.

FIG. **10**A is a diagram illustrating an exemplary instant messaging system **1000** that implements messenger-controlled applications. Messenger-controlled applications may also be referred to generally as instant messaging or IM applications. As described above in connection with FIG. **2**, instant messaging (IM) clients **202** are shown using the Internet for transferring data. Clients, referring to end user applications or instant messaging user applications, provide an implementation environment for messenger-controlled or IM applications. Conversation user interfaces (UI) **204** provide users with interfaces for creating, reviewing, editing, or modifying content to send between IM clients **202**. In the embodiment shown, an IM application may be executed between IM clients **202** using an IM message. This will be described in further detail below. As described in FIG. **2**, IM environments are stored in cache **206**. Also provided are IM application servers **1002** for implementing IM applications as will be described below. Also generally, an IM application is any application implemented in an instant messaging environment.

FIG. **10**B is a diagram illustrating an alternative embodiment of an instant messaging system. Also communicating over the Internet, IM clients **1006**, **1008** implement IM environments **1010**, **1012**. IM applications **1014**, **1016** can be implemented over IM environments **1010**, **1012**. An IM session is executed using conversation interfaces **1018**, **1020**.

FIG. **10**C is a diagram illustrating another embodiment of an instant messaging system using third-party servers. IM clients/networking layers **1022**, **1024** can implement IM environments **1026** and **1028**, also executing IM applications which may be stored on third-party servers **1030**, **1032**, **1034**, and **1036**. Similarly, an IM session may be controlled or managed using client conversation user interfaces, similar to the conversation interfaces **1018**, **1020** in FIG. **10**B.

FIG. **11** is a flow chart illustrating an exemplary process **1100** for handling message-controlled applications in an IM environment. In the preferred embodiment, data and information passed between users in the form of a message. Applications and data may also be considered "messages" and can be passed between IM clients **202** as "IM messages." In step **1102**, handlers in IM clients **202** receive messages. IM environments then evaluate the received message and determine an appropriate action for user and IM messages, as shown in step **1104**. In step **1106**, IM environments determine whether the message is an IM message. IM messages refer to messages useful in the execution and operation of IM applications. Conversely, messages between users (user messages) refer to communication messages between end users engaged in an IM session.

If an IM message is determined above, then an IM application (i.e. movie trailer, video, etc.) is executed. If the message is not an IM message, then IM environments determine whether the message is a user message in step **1110**. If the message is a user message, then the message is displayed via IM client **202** on conversation user interface **204**. If the message is neither an IM message nor a user message, then error handling is invoked, in step **1114**.

FIG. **12**A is a flow chart illustrating the control and execution of IM applications in an instant messaging sys-

tem, in accordance with one embodiment of the present invention. In step **1202**, IM client **202** (FIG. **2** or **10**) evaluates an IM message. From the IM message, the IM client **202** determines the application type (i.e., movie trailer, game, animated cartoon, advertisement, Flash presentation, etc.) in step **1204**. Using an identifier, the IM application is retrieved in step **1206**. In step **1208**, a decision is made as to whether a supporting application is required such as a media player (Real Player, Windows Media Player), content viewer (Adobe Illustrator, Reader, etc.), or other media-based display application. If required, the supporting application is launched in step **1210**. If a supporting application is not required or if the supporting application has been launched, then the IM application is implemented/executed in step **1212**. Alternative embodiments pertinent to the type of IM application implemented are shown in FIGS. **12B** through **12D**.

In FIG. **12B**, an exemplary flow chart for implementing a messenger-controlled IM application such as a movie trailer is shown. In step **1222**, the IM client **202** is directed to retrieve the identified IM application (e.g., a movie trailer) from a movie server such as IM application server **1002** (FIG. **10**). A movie trailer identifier is passed to the movie server in step **1224**. Based upon the identifier, a movie trailer or set of movie trailers may be retrieved in step **1226**. Upon retrieving the movie trailer, IM client **202** then returns to step **1208** (FIG. **12A**).

FIG. **12C** illustrates another exemplary flow chart for implementing a messenger-controlled IM application which yields financial data such as a stock quote, earnings indicator, or other financial data. In step **1232**, IM client **202** can be implemented using a server, database, RAID, storage disk, repository, etc. where the financial data application is stored. An identifier is passed, which identifies the specific financial data to be retrieved, such as a stock quote, public earnings figure, or other similar data in step **1234**. The financial data is retrieved in step **1236** using the identifier passed by IM client **202**.

FIG. **12D** illustrates another exemplary flow chart for implementing a messenger-controlled IM application such as an audio file (e.g., song, recording, etc.). IM client **202** is directed to an audio server storing files of music, songs, recordings, etc. in various formats such as WAV files, MPEG3 files, etc. Passing an identifier to the audio server (i.e., IM application server **1002**), IM client **202** can retrieve an audio file in step **1246**. Subsequently, IM client **202** continues at step **1208** of the process illustrated in FIG. **12A**.

FIG. **13** is a flow chart illustrating an exemplary IM application control process in accordance with one embodiment of the present invention. In step **1302**, a user can direct IM client **202** to implement a control feature which will control an IM application. In the context of FIG. **13**, a control message is similar to the above-discussed IM message. IM environment creates a control message in step **1304**. The control message is sent to IM client **202** which implements the requested IM application, in step **1306**.

FIG. **14A** is an exemplary user interface for controlling IM applications in accordance with one embodiment of the present invention. A computer monitor screen **1400** is shown, containing a history window **1402**. History window **1402** is similar to the history window **302** illustrated and discussed above in connection with FIG. **3**. Within history window **1402**, browser control elements and features are shown. For example, in the embodiment shown several framed areas are provided. In the embodiment shown, several I-frames are provided including a conversation area **1404**, content window **1406**, a link frame **1408** and an application area **1410**.

Control features such as those discussed in connection with the process flow chart of FIG. **13** may be included in any of the frames/windows shown. For example, in application area **1410**, a graphic link may be provided which, when directed, executes an application which plays a movie trailer (in accordance with the process described above for FIG. **12B**). In another example, a stock "ticker" may be played in application area **1410**, displaying company symbols and financial stock quotes. Users can control IM applications using an interface similar to that shown in FIG. **14A** (**1400**). In other embodiments, other techniques are used to implement control of messaging applications other than those described here and the embodiments listed above are intended to be neither comprehensive or inclusive of other applications that may be controlled from an IM environment.

FIG. **14B** illustrates another exemplary embodiment of application area **1410** for viewing content. A viewer window **1414** is provided for viewing or displaying content such as a movie trailer, film clip, animated story or advertisement, or video file in any of a number of formats including, but not limited to, MPEG4, Macromedia Flash, etc. A viewer button **1416** is provided which, when depressed, will launch an appropriate viewer/player, rendering the IM application/content for display to the user.

FIG. **14C** illustrates another exemplary embodiment of application area **1410** for entering a search term and performing a search based on the search term. In search term frame **1418**, a user may enter a search term. After depressing search button **1420**, an IM application is launched which executes the desired search based on the entered search term, returning and rendering the results in a frame on interface **1400**. The frame may include, but is not limited to those shown in FIG. **14A**. For example, returned search results may be included in content window **1406**, application area **1410**, or conversation area **1410**. Search results and terms may be placed in frames other than those shown in the embodiments described herein.

FIG. **14D** illustrates another exemplary embodiment of application area **1410** for executing enhanced functionality such as co-browsing in an IM environment. Co-browsing, or two or more users browsing for online content while engaged in an IM environment, is provided for as a messenger-controlled IM application. URL/domain frame **1422** is configured to received a URL or domain name for pointing an IM application such as a web browser to a particular Internet or website. In one embodiment, the IM environment may launch a web browser once the co-browse button **1424** is depressed. Once a co-browsing session has been initiated, two or more users will be directed to the same Internet or website based upon the destination URL/domain name entered into URL/domain frame **1422**. Browsers such as those provided by Microsoft (Internet Explorer), Netscape (Navigator), etc. may be used. Other internet browsing applications are used in other embodiments.

Control of IM applications such as co-searching, co-browsing, networked games, shared videos or audio files may be implemented in an automatic, semi-automatic, or manual manner. Information related to the control of IM applications uses IM messages which, when passed between IM clients, causes particular actions/interactions to occur between two or more IM users. The IM messages enable and control IM applications, which is widely varied in terms of functionality and user interactivity. IM applications may include co-searching, co-browsing, site navigation, co-drawing, media sharing (e.g., video, music, animation, photos, etc.), games, and activities in other embodiments. The IM applications can be hosted on servers communicating with the instant messaging clients, which may be operated by a variety of entities such as a content developer, hosted service, service provider, etc.

Procedures and techniques for messenger-controlled applications in instant messaging environments have been described. A number of exemplary environments that may

be created using these procedures and techniques have also been shown. In addition to being used in instant messaging environments between two users, the techniques described herein may also apply to instant messaging environments that include one, three, or more users.

Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. It should be noted that there are many alternative ways of implementing both the process and apparatus of the present invention. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

What is claimed is:

1. A machine-implemented method of controlling an application, said method comprising:

in response to receiving user input at a first instant messaging client and a second instant messaging client, generating user messages at the first instant messaging client and the second instant messaging client, wherein the first instant messaging client has a first conversation user interface and the second instant messaging client has a second conversation user interface;

exchanging the user messages, via an instant messaging system, between the first instant messaging client and the second instant messaging client;

displaying a conversation in the first conversation user interface and the second conversation user interface, wherein said conversation is based on the user messages transferred via the instant messaging system;

in response to user input at the first instant messaging client, generating a control message at the first instant messaging client, wherein the control message indicates an identifier associated with an application that is retrieved and executed from the first instant messaging client

transferring the control message, via the instant messaging system, between the first instant messaging client and the second instant messaging client; and

in response to receiving the control message, causing results of executing the application as specified by the control message to be reflected in a display provided by the second instant messaging client.

2. A method as recited in claim 1, further comprising:

determining whether a message received at the second client is one of the user messages or the control message.

3. A method as recited in claim 1 wherein the step of causing results of executing the application as specified by the control message to be reflected in a display provided by the second instant messaging client comprises playing results of executing the application in a player.

4. A method as recited in claim 1 wherein the step of causing results of executing the application as specified by the control message to be reflected in a display provided by the second instant messaging client comprises displaying the results of executing the application in a viewer.

5. A method as recited in claim 1 further comprising arbitrating data received from a third instant messaging client based on executing the application.

6. A method as recited in claim 1 further comprising, prior to generating the control message, receiving a selection of the application from a list of instant messaging applications at the first instant messaging client.

7. A method as recited in claim 1 further comprising causing results of executing the application to be reflected in a display provided by the first instant messaging client.

8. A computer-readable storage medium carrying one or more sequences of instructions, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of:

in response to receiving user input at a first instant messaging client and a second instant messaging client, generating user messages at the first instant messaging client and the second instant messaging client, wherein the first instant messaging client has a first conversation user interface and the second instant messaging client has a second conversation user interface;

exchanging the user messages, via an instant messaging system, between the first instant messaging client and the second instant messaging client;

displaying a conversation in the first conversation user interface and the second conversation user interface, wherein said conversation is based on the user messages transferred via the instant messaging system;

in response to user input at the first instant messaging client, generating a control message at the first instant messaging client, wherein the control message indicates an identifier associated with an application that is retrieved and executed from the first instant messaging client

transferring the control message, via the instant messaging system, between the first instant messaging client and the second instant messaging client; and

in response to receiving the control message, causing results of executing the application as specified by the control message to be reflected in a display provided by the second instant messaging client.

9. A computer-readable storage medium as recited in claim 8, further comprising instructions which, when executed by the one or more processors, cause the one or more processors to perform the step of determining whether a message received at the second client is one of the user messages or the control message.

10. A computer-readable storage medium as recited in claim 8 wherein the instructions for performing the step of causing results of executing the application as specified by the control message to be reflected in a display provided by the second instant messaging client further comprise instructions for playing results of executing the application in a player.

11. A computer-readable storage medium as recited in claim 8 wherein the instructions for performing the step of causing results of executing the application as specified by the control message to be reflected in a display provided by the second instant messaging client further comprise instructions for displaying the results of executing the application in a viewer.

12. A computer-readable storage medium as recited in claim 8 further comprising instructions which, when executed by the one or more processors, cause the one or more processors to perform the step of arbitrating data received from a third instant messaging client based on executing the application.

13. A computer-readable storage medium as recited in claim 8 further comprising instructions which, when executed by the one or more processors, cause the one or more processors to perform the step of:

prior to generating the control message, receiving a selection of the application from a list of instant messaging applications at the first instant messaging client.

14. A computer-readable storage medium as recited in claim 8 further comprising instructions which, when executed by the one or more processors, cause the one or more processors to perform the step of causing results of executing the application to be reflected in a display provided by the first instant messaging client.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/589,155	05/04/2007	Claus Pedersen	884A.0148.U1(US)	6810

29683 7590 11/25/2008
HARRINGTON & SMITH, PC
4 RESEARCH DRIVE, Suite 202
SHELTON, CT 06484-6212

EXAMINER	
LEE, CHUN KUAN	
ART UNIT	PAPER NUMBER
2181	

MAIL DATE	DELIVERY MODE
11/25/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

SHORTENED 12/29/08 - 1 1mo
DUE DATE
PAPER DATED 11/25/08
OA FINAL 2/28/09
MSG PT DWG
APPEAL ISSUE FEE
OTHER Advisory Action

RECEIVED
NOV 28 2008

Advisory Action Before the Filing of an Appeal Brief	Application No. 10/589,155	Applicant(s) PEDERSEN ET AL.	
	Examiner Chun-Kuan Lee	Art Unit 2181	

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

THE REPLY FILED 27 October 2008 FAILS TO PLACE THIS APPLICATION IN CONDITION FOR ALLOWANCE.

1. ☒ The reply was filed after a final rejection, but prior to or on the same day as filing a Notice of Appeal. To avoid abandonment of this application, applicant must timely file one of the following replies: (1) an amendment, affidavit, or other evidence, which places the application in condition for allowance; (2) a Notice of Appeal (with appeal fee) in compliance with 37 CFR 41.31; or (3) a Request for Continued Examination (RCE) in compliance with 37 CFR 1.114. The reply must be filed within one of the following time periods:
   a) ☐ The period for reply expires _____ months from the mailing date of the final rejection.
   b) ☒ The period for reply expires on: (1) the mailing date of this Advisory Action, or (2) the date set forth in the final rejection, whichever is later. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of the final rejection.

   Examiner Note: If box 1 is checked, check either box (a) or (b). ONLY CHECK BOX (b) WHEN THE FIRST REPLY WAS FILED WITHIN TWO MONTHS OF THE FINAL REJECTION. See MPEP 706.07(f).

Extensions of time may be obtained under 37 CFR 1.136(a). The date on which the petition under 37 CFR 1.136(a) and the appropriate extension fee have been filed is the date for purposes of determining the period of extension and the corresponding amount of the fee. The appropriate extension fee under 37 CFR 1.17(a) is calculated from: (1) the expiration date of the shortened statutory period for reply originally set in the final Office action; or (2) as set forth in (b) above, if checked. Any reply received by the Office later than three months after the mailing date of the final rejection, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

NOTICE OF APPEAL

2. ☐ The Notice of Appeal was filed on _____. A brief in compliance with 37 CFR 41.37 must be filed within two months of the date of filing the Notice of Appeal (37 CFR 41.37(a)), or any extension thereof (37 CFR 41.37(e)), to avoid dismissal of the appeal. Since a Notice of Appeal has been filed, any reply must be filed within the time period set forth in 37 CFR 41.37(a).

AMENDMENTS

3. ☐ The proposed amendment(s) filed after a final rejection, but prior to the date of filing a brief, will not be entered because
   (a) ☐ They raise new issues that would require further consideration and/or search (see NOTE below);
   (b) ☐ They raise the issue of new matter (see NOTE below);
   (c) ☐ They are not deemed to place the application in better form for appeal by materially reducing or simplifying the issues for appeal; and/or
   (d) ☐ They present additional claims without canceling a corresponding number of finally rejected claims.

   NOTE: _____. (See 37 CFR 1.116 and 41.33(a)).
4. ☐ The amendments are not in compliance with 37 CFR 1.121. See attached Notice of Non-Compliant Amendment (PTOL-324).
5. ☒ Applicant's reply has overcome the following rejection(s): Please see Continuation Sheet below.
6. ☐ Newly proposed or amended claim(s) _____ would be allowable if submitted in a separate, timely filed amendment canceling the non-allowable claim(s).
7. ☒ For purposes of appeal, the proposed amendment(s): a) ☐ will not be entered, or b) ☒ will be entered and an explanation of how the new or amended claims would be rejected is provided below or appended.
   The status of the claim(s) is (or will be) as follows:
   Claim(s) allowed: _____.
   Claim(s) objected to: _____.
   Claim(s) rejected: *1,3-10,12-28,34-41,43 and 44*.
   Claim(s) withdrawn from consideration: _____.

AFFIDAVIT OR OTHER EVIDENCE

8. ☐ The affidavit or other evidence filed after a final action, but before or on the date of filing a Notice of Appeal will not be entered because applicant failed to provide a showing of good and sufficient reasons why the affidavit or other evidence is necessary and was not earlier presented. See 37 CFR 1.116(e).
9. ☐ The affidavit or other evidence filed after the date of filing a Notice of Appeal, but prior to the date of filing a brief, will not be entered because the affidavit or other evidence failed to overcome all rejections under appeal and/or appellant fails to provide a showing a good and sufficient reasons why it is necessary and was not earlier presented. See 37 CFR 41.33(d)(1).
10. ☐ The affidavit or other evidence is entered. An explanation of the status of the claims after entry is below or attached.

REQUEST FOR RECONSIDERATION/OTHER

11. ☒ The request for reconsideration has been considered but does NOT place the application in condition for allowance because: Please see Continuation Sheet below.
12. ☐ Note the attached Information *Disclosure Statement*(s). (PTO/SB/08) Paper No(s). _____
13. ☐ Other: _____.

/Alford W. Kindred/
Supervisory Patent Examiner, Art Unit 2181

Continuation of 5. Applicant's reply has overcome the following rejection(s): Rejection of claims 16, 21, 34 and 43-44 under 35 U.S.C. 112 second paragraph.

In response to applicant's arguments (on page 11, 3rd paragraph) with regard to the independent claim 1 rejected under 35 U.S.C. 103(a) that the combination of the references does not teach/suggest the claimed feature that the command effectively commands 'do something on this identified data' but does not specify what should be done, because Rao discloses the use of "conventional" command, as the enhancement commands operate in a conventional way by specifying a particular executable rather than specifying execution of an unidentified executable; applicant's arguments have fully been considered, but are not found to be persuasive.

The examiner respectfully disagrees, because Rao's command itself is not conventional, because the command is an enhancement command, wherein such command does not previously exist/conventional to SyncML technology. As for the claim feature of the command that commands 'do something on this identified data' but does not specify what should be done, the examiner is relying on SyncML Meta-Information DTD's and Szeto's teaching.

In response to applicant's argument (on page 11, last paragraph to page 12, 1st paragraph) that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and In re Jones, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, Rao's teaching is operating in accordance to the SyncML technology, therefore it would then be motivated to combine SyncML operational specification (i.e. SyncML reference) into Rao so that Rao's teaching can properly operate in the SyncML environment.

In response to applicant's arguments (on page 12, 2nd paragraph to page 13, 3rd paragraph) with regard to the independent claim 1 rejected under 35 U.S.C. 103(a) that the combination of references does not teach/suggest the claimed feature that the command effectively commands 'do something on this identified data' but does not specify what should be done, because Szeto's control message specifies an IM application to be retrieved using an identifier; applicant's arguments have fully been considered, but are not found to be persuasive.

The examiner respectfully disagrees, because with regard to the received IM message including the identifier, the initial application initiated for the received IM message would be the corresponding IM application (e.g. well known MSN messenger), and subsequent to examining the IM message and the corresponding identifier (e.g. metadata) will the supporting application (e.g. application other then the initial initiated application) be initiated; therefore, the supporting application is identified using the identifier metadata in the IM message. Furthermore, the examiner expressly relied on SyncML Meta-Information DTD for the teaching corresponding to metadata for the SyncML environment.

In response to applicant's arguments (on page 13, 4th paragraph) with regard to the independent claim 1 rejected under 35 U.S.C. 103(a), wherein applicant appears to be arguing that Szeto is nonanalogous art, because Szeto clearly has no relation to SyncML and there is no disclosure of using metadata to determine content type of data; applicant's arguments have fully been considered, but are not found to be persuasive.

Please note that it has been held that a prior art reference must either be in the field of applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the applicant was concerned, in order to be relied upon as a basis for rejection of the claimed invention. See In re Oetiker, 977 F.2d 1443, 24 USPQ2d 1443 (Fed. Cir. 1992).

In this case, not only is Szeto in the field of applicant's endeavor, as Szeto's invention is implement in a network environment (e.g. SyncML also operate in a network environment); furthermore, Szeto is reasonably pertinent to the particular problem with which the applicant was concerned, which is to identify the corresponding application using metadata (e.g. identifier of the IM message).

Applicant's amendments do not change how the examiner is relying on the prior art of record for the teaching of the claimed features. And, in responding to all applicant's arguments, the examiner will maintain his position and the current rejection of record.